Mathematics of Information & Entropy

Volume II: Coding Theory & Data Compression

Block codes, linear codes, turbo/LDPC, arithmetic coding, rate-distortion theory, and Kolmogorov complexity

---

Every derivation step by step

Every symbol defined

Explicit numerical examples

Geometric and combinatorial intuitions

Mathematics of Information & Entropy Series

Contents

1.1 The Coding Problem: Combinatorial Formulation 4

1.2 Sphere-Packing Bounds: Hamming and Plotkin 5

1.3 Hamming Codes: Construction and Optimality 7

1.4 The Gilbert-Varshamov Bound and Probabilistic Existence Arguments 8

1.5 Weight Enumerators and the MacWilliams Identity 9

2.1 Finite Fields: Construction and Arithmetic 11

2.2 Linear Codes: Generator and Parity-Check Matrices 12

2.3 Cyclic Codes and the Polynomial Representation 13

2.4 Reed-Solomon Codes: Construction and Minimum Distance 13

2.5 The Berlekamp-Massey Algorithm 14

2.6 BCH Codes and Their Distance Properties 15

3.1 Low-Density Parity-Check Codes 16

3.2 The Sum-Product Algorithm (Belief Propagation) 16

3.3 Density Evolution and Capacity-Achieving LDPC Sequences 18

3.4 Turbo Codes and the Turbo Principle 19

4.1 Prefix-Free Codes and the Huffman Algorithm 21

4.2 Arithmetic Coding 22

4.3 Universal Source Coding: The Lempel-Ziv Family 24

4.4 Rate-Distortion Theory 25

4.5 The Blahut-Arimoto Algorithm for Rate-Distortion 26

5.1 Computability Prerequisites 28

5.2 Basic Properties of Kolmogorov Complexity 28

5.3 The Incompressibility Method 29

5.4 The Universal Prior and Solomonoff Induction 30

5.5 Connections to Shannon Entropy: The AEP Revisited 30

Numerical Tour: Part 1 — Sphere-Packing and Weight Enumerators 32

Numerical Tour: Part 2 — Reed-Solomon Encoding and Decoding 32

Numerical Tour: Part 3 — LDPC Belief Propagation on the BEC 33

Numerical Tour: Part 4 — Arithmetic Coding and Rate-Distortion 33

Formula Reference: Part 1 — Block Codes 34

Formula Reference: Part 2 — Linear and Cyclic Codes 34

Formula Reference: Part 3 — LDPC and Turbo Codes 35

Formula Reference: Part 4 — Data Compression 35

Formula Reference: Part 5 — Kolmogorov Complexity 35

Computational Notes: Algorithms and Implementation 37

Glossary of Key Terms and Symbols 40

A Proof of the MacWilliams Identity 42

B Proof of Shannon's Rate-Distortion Theorem (Converse) 43

C Linear Programming Bounds (Delsarte's Method) 43

D Turbo Code Distance Properties and the Error Floor 44

E Connections to Cryptography and Information Security 44

F Open Problems in Coding Theory and Compression 45

Cross-Volume Connections and Methodological Synthesis 46

Extended Case Study: End-to-End Design of a Data Storage System 47

6.1 The Polarization Phenomenon 51

6.2 Polar Code Construction and Encoding 53

6.3 Successive Cancellation Decoding 54

6.4 Successive Cancellation List Decoding 55

6.5 Polar Codes in 5G NR 55

7.1 The Slepian-Wolf Theorem 57

7.2 The Wyner-Ziv Theorem: Lossy Compression with Side Information 58

7.3 Successive Refinement 58

7.4 Multiple Description Coding 59

8.1 The AWGN Channel and Gaussian Capacity 60

8.2 Water-Filling: Capacity of Parallel Gaussian Channels 60

8.3 The Multiple-Access Channel **61**

8.4 Fountain Codes: LT Codes and Raptor Codes **62**

8.5 Network Coding and Information Flow **63**

9.1 Beyond Asymptotic Capacity: The PPV Framework **64**

9.2 Achievability and Converse Bounds in the PPV Framework **65**

9.3 Short-Packet Communications in 5G URLLC **65**

Supplementary Derivations **67**

Numerical Tour: Part 5 — Polar Code Construction for BEC **70**

Numerical Tour: Part 6 — Finite Blocklength Bounds for AWGN **70**

Numerical Tour: Part 7 — Slepian-Wolf and Distributed Compression **71**

Numerical Tour: Part 8 — Reed-Muller Codes **71**

Theoretical Complements: The Geometry of Codes **73**

Information Inequalities and Coding Theory **75**

Problems and Exercises **76**

10.1 Lattices: Definitions and Basic Properties **78**
Lattice Fundamentals . . . 78
Construction A: Codes to Lattices . . . 78
Lattice Quantisation and the MMSE Distortion . . . 79
Nested Lattice Codes for the AWGN Channel . . . 79

10.2 The Sphere-Packing Bound for Lattices **79**
Minkowski–Hlawka Theorem . . . 79
The Elias–Bassalygo Comparison . . . 80

10.3 Lattice Coding for Gaussian Multiple-Access Channels **80**
Integer-Forcing Receivers . . . 80
Compute-and-Forward Protocol . . . 80

10.4 The Relationship Between Lattice Codes and Polar Codes **81**

11.1 The Flash Memory Channel **82**
NAND Flash Physics . . . 82
Rank Modulation . . . 82
Rewriting Codes and WOM . . . 83

11.2 ECC for NAND Flash in Practice **83**
Soft-Decision Decoding for Flash . . . 83
The Flash Channel as an Information Channel . . . 83

11.3 DNA Data Storage: Information Theory **84**
The DNA Storage Channel . . . 84

Information Capacity of DNA Storage . . . 84
Coding Schemes for DNA Storage . . . 84
Insertion-Deletion-Substitution Codes for DNA . . . 85

11.4 Information-Theoretic Limits of DNA Storage **85**
The DNA Storage Channel Capacity . . . 85
The Shotgun DNA Sequencing Problem . . . 85

Supplementary Derivations (Continued) **87**

Numerical Tour: Part 9 — Lattice Coding for the Gaussian Channel **89**

Numerical Tour: Part 10 — DNA Storage Rate Calculation **89**

Numerical Tour: Part 11 — Weight Enumerator Bounds via LP **90**

Formula Reference: Parts 6–11 Supplement **90**

I Asymptotic Analysis: The Random Coding Exponent **91**

J Proofs of Selected Results from Parts 6–11 **91**

12.1 The Index Coding Problem **93**
Problem Formulation . . . 93
Connection to Rank and Graph Theory . . . 93

12.2 Coded Caching: The Maddah-Ali–Niesen Scheme **93**
The Coded Caching Model . . . 94
Information-Theoretic Optimality . . . 95

12.3 Connection to Interference Alignment **95**

13.1 Qubits and Quantum Channels **96**
Quantum States . . . 96
Quantum Channels . . . 96
Von Neumann Entropy and Quantum Information Quantities . . . 96

13.2 Quantum Error Correction: Stabiliser Codes **97**
The Quantum Error Correction Conditions . . . 97
Stabiliser Codes . . . 97
The $[[7, 1, 3]]$ Steane Code . . . 97

13.3 The Quantum Shannon Theorem **98**
Quantum Channel Capacity . . . 98
Superdense Coding and Quantum Teleportation . . . 98

13.4 The Hashing Bound and Stabiliser Code Design **99**
The Quantum Hamming Bound . . . 99
Degenerate vs. Non-Degenerate Codes . . . 99

Numerical Tour: Part 12 — Coded Caching Gain Calculation **100**

Numerical Tour: Part 13 — Steane Code Error Correction **100**

Formula Reference: Parts 12–13 100

K The Information-Geometric Perspective on Codes and Algorithms 102

L Selected Problems with Solutions 103

Closing Synthesis: Unifying Themes 104

M Advanced Topics in Decoding Complexity 106

N Lossy Compression Standards: JPEG to Neural Codecs 107

Final Formula Compendium 108

Preface to Volume II

Scope and motivation. Volume I established the foundational trinity of information theory: entropy as a measure of uncertainty, mutual information as a measure of statistical dependence, and channel capacity as the fundamental limit on reliable communication. It showed, via Shannon's two landmark theorems, that these quantities are not mere mathematical curiosities but are operationally achievable: there exist source codes whose rate approaches entropy arbitrarily closely, and there exist channel codes whose rate approaches capacity with error probability tending to zero.

What Volume I *did not* do was construct these codes explicitly, nor analyze their complexity and performance in finite blocklengths. That is the task of Volume II. Coding theory is the discipline that takes Shannon's existence proofs and asks: *how do we build codes that actually work, and how do we decode them efficiently?* Data compression theory asks the complementary question: given a source model, what is the minimum number of bits needed to represent a message, and how do we construct encoders and decoders that approach this limit?

This volume develops three interconnected threads. The first is *algebraic coding theory*: the construction and analysis of block codes and linear codes using the algebra of finite fields, culminating in Reed-Solomon codes and their decoding algorithms. The second is *iterative coding*: the modern turbo and LDPC paradigm, where belief propagation on sparse graphical representations achieves performance within a fraction of a decibel of the Shannon limit. The third is *data compression*: arithmetic coding as a universal near-optimal scheme for lossless compression, rate-distortion theory as the information-theoretic limit for lossy compression, and Kolmogorov complexity as an absolute, non-probabilistic notion of information content.

Structure. The volume is organized into five parts. Part 1 lays the combinatorial and algebraic foundations of block codes: Hamming distance, sphere-packing bounds, and the Singleton bound. Part 2 develops the theory of linear codes over finite fields, including parity-check matrices, dual codes, the MacWilliams identity, and the complete family of Reed-Solomon codes with Berlekamp-Massey and Euclidean decoding. Part 3 treats modern iterative codes: LDPC codes via Tanner graphs, the sum-product algorithm for belief propagation, density evolution for performance analysis, and turbo codes with their turbo principle. Part 4 is devoted to data compression in depth: Huffman coding optimality proofs, arithmetic coding with precision analysis, the Lempel-Ziv family of universal codes, and rate-distortion theory from its operational definition through the Blahut-Arimoto algorithm for rate-distortion functions. Part 5 treats Kolmogorov complexity: the universal semi-measure, the incompressibility method, and connections to computational complexity and statistical inference.

Prerequisites and connections. This volume presupposes familiarity with the material in Volume I, especially entropy, mutual information, the asymptotic equipartition property, the method of types, and the channel coding theorem. It also uses basic concepts from linear algebra (vector spaces, matrix rank) and from abstract algebra (fields, polynomials, group theory), which are developed from scratch in the text as needed. The notation follows Volume I exactly. Volume III will build on the exponential family framework developed in Section 2.6 of Volume I and the structure of divergences to develop information geometry; readers of Volume II will find the Reed-Solomon codes appearing again in the context of algebraic geometry codes, and the LDPC density evolution reappearing as a statement about free energy in a graphical model.

A note on computation. Data compression and coding are inherently computational disciplines, and this volume includes full algorithm specifications for every major scheme dis-

cussed: the Berlekamp-Massey algorithm, belief propagation for LDPC, arithmetic coding with interval arithmetic, and the Blahut-Arimoto algorithm for rate-distortion functions. Where useful, pseudocode is provided in a mathematically transparent form. The goal is not software engineering but mathematical clarity: every algorithm is justified by the theorem it implements.

Part 1 — Block Codes and Combinatorial Bounds

Hamming distance and the metric structure of binary strings. Sphere-packing (Hamming) bound, the Singleton bound, and the Plotkin bound. Perfect codes: Hamming codes and the Golay code. Gilbert-Varshamov bound. Weight enumerators and the MacWilliams identity (preview). The binary entropy function and its role in asymptotic bounds.

1.1 The Coding Problem: Combinatorial Formulation

A communication system faces the following problem. A sender wishes to transmit one of M distinct messages across a noisy channel that may corrupt some symbols of the transmitted codeword. The receiver observes the corrupted codeword and must determine which of the M messages was sent. We model the channel as a *binary symmetric channel* (BSC) with crossover probability $\epsilon \in (0, 1/2)$: each transmitted bit is independently flipped to the opposite value with probability ϵ, and transmitted correctly with probability $1 - \epsilon$.

The fundamental insight of coding theory is that introducing *redundancy* into the transmitted message can make reliable communication possible even when $\epsilon > 0$. A *code* of *block length* n is a set $\mathcal{C} \subseteq \{0,1\}^n$ of *codewords*. The *rate* of the code is

$$R = \frac{\log_2 M}{n} = \frac{\log_2 |\mathcal{C}|}{n} \text{ bits per channel use.}$$

The combinatorial structure that determines error-correcting capability is the *Hamming distance*.

Hamming Distance and Hamming Weight

For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, the *Hamming distance* is

$$d_H(\mathbf{x}, \mathbf{y}) \triangleq |\{i : x_i \neq y_i\}| = \sum_{i=1}^{n} \mathbf{1}\{x_i \neq y_i\},$$

the number of coordinate positions in which $\mathbf{x}$ and $\mathbf{y}$ differ. The *Hamming weight* of $\mathbf{x}$ is $w_H(\mathbf{x}) \triangleq d_H(\mathbf{x}, \mathbf{0})$, the number of 1s in $\mathbf{x}$. The *minimum distance* of a code $\mathcal{C}$ is

$$d_{\min}(\mathcal{C}) \triangleq \min_{\mathbf{c} \neq \mathbf{c}' \in \mathcal{C}} d_H(\mathbf{c}, \mathbf{c}').$$

Hamming distance is a metric. We verify the three metric axioms. Non-negativity and symmetry are immediate from the definition. For the triangle inequality: for any $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0,1\}^n$,

$$d_H(\mathbf{x}, \mathbf{z}) = \sum_i \mathbf{1}\{x_i \neq z_i\} \leq \sum_i \mathbf{1}\{x_i \neq y_i\} + \sum_i \mathbf{1}\{y_i \neq z_i\} = d_H(\mathbf{x}, \mathbf{y}) + d_H(\mathbf{y}, \mathbf{z}),$$

since any position where $x_i \neq z_i$ must satisfy $x_i \neq y_i$ or $y_i \neq z_i$ (or both), because $\{0,1\}$ has only two elements.

Error detection and correction in terms of minimum distance. A code $\mathcal{C}$ with minimum distance d can:

- *detect* up to $d - 1$ errors: if at most $d - 1$ bits are flipped, the received word cannot be a codeword different from the transmitted one (since any two codewords differ in at least d positions).

- *correct* up to $\lfloor (d-1)/2 \rfloor$ errors via *minimum-distance decoding*: decode to the nearest codeword in Hamming distance. If at most $t = \lfloor (d-1)/2 \rfloor$ errors occur, the received word is closer to the transmitted codeword than to any other, because the Hamming balls of radius t around distinct codewords are disjoint (since $2t < d$).

We write a code with parameters block length n, $M = 2^k$ codewords (for integer k), and minimum distance d as an (n, k, d) code. The rate is k/n bits per channel use.

Example — The repetition code $(3, 1, 3)$

Encode $0 \mapsto 000$ and $1 \mapsto 111$. Then $n = 3$, $k = 1$ (rate $1/3$), $d_{\min} = 3$. Minimum-distance decoding corrects all single-bit errors (since $\lfloor (3-1)/2 \rfloor = 1$):

Received word	Decoded message
$000, 001, 010, 100$	0
$111, 110, 101, 011$	1

This is a perfect code: the $2^1 = 2$ balls of radius 1 around the codewords $\{000, 111\}$ together cover all $2^3 = 8$ binary strings of length 3 (each ball has $1 + 3 = 4$ elements, and $2 \times 4 = 8$). No codeword is wasted.

1.2 Sphere-Packing Bounds: Hamming and Plotkin

The key combinatorial question of coding theory is: for given n and d, what is the maximum size $A(n, d)$ of a binary code with minimum distance d? Upper bounds on $A(n, d)$ constrain what codes are achievable; lower bounds guarantee that good codes exist. In this section we derive two fundamental upper bounds: the Hamming (sphere-packing) bound and the Plotkin bound.

The Hamming Sphere-Packing Bound

Define the *Hamming ball* of radius r centered at $\mathbf{x} \in \{0,1\}^n$ as

$$B(\mathbf{x}, r) \triangleq \{\mathbf{y} \in \{0,1\}^n : d_H(\mathbf{x}, \mathbf{y}) \le r\}.$$

Its size is $|B(\mathbf{x}, r)| = V(n, r)$, the *volume* of a Hamming ball:

$$V(n, r) = \sum_{j=0}^{r} \binom{n}{j}.$$

This is because to construct a word at Hamming distance exactly j from $\mathbf{x}$, we choose which j of the n positions to flip, giving $\binom{n}{j}$ choices; summing over $j = 0, \dots, r$ gives the ball volume.

Hamming Bound (Sphere-Packing Bound)

For any binary (n, k, d) code with $d = 2t + 1$ (odd),

$$2^k \cdot V(n, t) \le 2^n, \qquad \text{equivalently} \qquad k \le n - \log_2 V(n, t).$$

Proof. The balls of radius $t = \lfloor (d-1)/2 \rfloor$ around distinct codewords are disjoint (since any two codewords differ in at least $d = 2t + 1$ positions, so a point within distance t of one codeword is at distance at least $d - t = t + 1 > t$ from any other). The union of these 2^k disjoint balls must be contained in $\{0,1\}^n$, which has 2^n elements. Therefore $2^k \cdot V(n, t) \le 2^n$. ∎

Asymptotic form. Let $\delta = d/n$ be the *relative minimum distance* and $\tau = t/n \approx \delta/2$. As $n \to \infty$ with δ and $R = k/n$ fixed, the Hamming bound becomes

$$R \leq 1 - h_b(\tau) = 1 - h_b(\delta/2),$$

where $h_b(p) = -p \log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function. This follows from Stirling's approximation: $\frac{1}{n}\log_2 V(n, \lfloor \tau n \rfloor) \to h_b(\tau)$ as $n \to \infty$ for $\tau \in [0, 1/2]$.

Example — Hamming bound for $d = 3$, binary code

A binary $(n, k, 3)$ code must satisfy $k \leq n - \log_2(1+n)$, since $V(n,1) = 1 + n$. For $n = 7$: $k \leq 7 - \log_2 8 = 7 - 3 = 4$. The Hamming code $[7, 4, 3]$ achieves this bound with equality: it is a *perfect code*. For $n = 15$: $k \leq 15 - \log_2 16 = 11$. The $[15, 11, 3]$ Hamming code is again perfect.

The Plotkin Bound

The Hamming bound is effective when d is small relative to n. For $d > n/2$, the Plotkin bound gives a stronger constraint.

Plotkin Bound

If $d > n/2$, then

$$A(n, d) \leq 2d.$$

More precisely, if $\mathcal{C}$ is a binary code with $|\mathcal{C}| = M$ and minimum distance d, then $M(2d - n) \leq 2d$.

Proof. Count the sum $S = \sum_{\mathbf{c} \neq \mathbf{c}'} d_H(\mathbf{c}, \mathbf{c}')$ in two ways. First, since every pair of distinct codewords has distance at least d, and there are $\binom{M}{2}$ pairs, we get $S \geq \binom{M}{2} d = \frac{M(M-1)}{2} d$.

Second, count contributions position by position. For coordinate i, let a_i be the number of codewords with a 0 in position i, and $b_i = M - a_i$ the number with a 1. The number of pairs that differ in position i is $a_i b_i$. Therefore

$$S = \sum_{i=1}^{n} a_i b_i = \sum_{i=1}^{n} a_i (M - a_i).$$

Since $a_i(M - a_i)$ is maximized at $a_i = M/2$ (by AM-GM) with maximum value $M^2/4$, we get $S \leq n \cdot M^2/4$.

Combining both bounds:

$$\frac{M(M-1)}{2} d \leq S \leq \frac{nM^2}{4} \implies \frac{(M-1)d}{2} \leq \frac{nM}{4} \implies 2d(M-1) \leq nM \implies M(2d-n) \leq 2d.$$

If $d > n/2$ then $2d - n > 0$, so $M \leq 2d/(2d-n)$. To get the clean bound $M \leq 2d$, observe that $2d/(2d-n) \leq 2d$ iff $1 \leq 2d - n$, i.e. $d \geq (n+1)/2$, which holds since $d > n/2$ and d is an integer. ■

The Singleton Bound and MDS Codes

A third fundamental bound applies to *q-ary* codes, where the alphabet has q symbols.

Singleton Bound

For a q-ary (n, k, d) code (with q^k codewords),

$$k \leq n - d + 1, \qquad \text{equivalently} \qquad d \leq n - k + 1.$$

The quantity $n-k+1$ is called the *Singleton bound* and a code achieving it with equality is called a *maximum-distance separable* (*MDS*) *code*.

Proof. Consider puncturing the code by deleting the last $d-1$ coordinates from every codeword. This maps each codeword $\mathbf{c} \in \mathcal{C}$ to a string in $\{0, \dots, q-1\}^{n-d+1}$. Claim: this map is injective. If two codewords $\mathbf{c}, \mathbf{c}'$ agree on all $n-d+1$ remaining coordinates, they differ in at most $d-1 < d$ positions (among the deleted ones), contradicting $d_H(\mathbf{c}, \mathbf{c}') \geq d$. So the q^k distinct codewords inject into a set of size q^{n-d+1}, giving $q^k \leq q^{n-d+1}$, i.e. $k \leq n - d + 1$. ■

The Singleton bound is remarkable because it is simultaneously a constraint on *rate* and *distance*: to achieve a large minimum distance d, the number of information symbols k cannot be too large. MDS codes saturate this tradeoff perfectly. As we will see in Part 2, Reed-Solomon codes are the canonical family of MDS codes.

1.3 Hamming Codes: Construction and Optimality

Having established the sphere-packing bound, we now construct a family of codes that achieves it with equality: the *Hamming codes*. These are the simplest non-trivial perfect codes and serve as the gateway to the linear algebra machinery of Part 2.

Binary Hamming Code

For integer $r \geq 2$, the *binary Hamming code* $\mathcal{H}_r$ has parameters

$$n = 2^r - 1, \qquad k = 2^r - 1 - r, \qquad d_{\min} = 3.$$

Its parity-check matrix $\mathbf{H} \in \{0,1\}^{r \times n}$ has as its i-th column the binary representation of the integer i, for $i = 1, 2, \dots, 2^r - 1$.

Construction. A codeword is any binary vector $\mathbf{c} \in \{0,1\}^n$ satisfying $\mathbf{Hc} = \mathbf{0}$ (over $\mathbb{F}_2$, i.e. with arithmetic modulo 2). This is a system of r homogeneous linear equations in $n = 2^r - 1$ unknowns, so the solution space has dimension $k = n - r = 2^r - 1 - r$ (assuming $\mathbf{H}$ has rank r, which holds since its columns include the r standard basis vectors $e_1, \dots, e_r$).

Minimum distance is 3.

- $d_{\min} \geq 3$: no single column of $\mathbf{H}$ is $\mathbf{0}$ (columns are nonzero integers 1 through $2^r - 1$), so no weight-1 vector is a codeword ($d_{\min} > 1$). No two columns are equal (all $2^r - 1$ distinct binary strings of length r appear exactly once), so no weight-2 vector is a codeword (a weight-2 codeword would require two columns to sum to $\mathbf{0}$ over $\mathbb{F}_2$, i.e. to be equal). Hence $d_{\min} \geq 3$.
- $d_{\min} \leq 3$: since every nonzero binary string of length r appears as a column of $\mathbf{H}$, in particular the three strings $e_1 = (1, 0, \dots, 0)^\top$, $e_2 = (0, 1, 0, \dots, 0)^\top$, and $e_1 + e_2 = (1, 1, 0, \dots, 0)^\top$ all appear. The sum $e_1 + e_2 = e_1 + e_2$ (a column of $\mathbf{H}$) shows that the three columns corresponding to integers 1, 2, 3 satisfy $\text{column}_1 + \text{column}_2 = \text{column}_3$, so the codeword with 1s in positions 1, 2, and 3 (and 0s elsewhere) has weight 3 and lies in $\mathcal{H}_r$. Thus $d_{\min} \leq 3$.

Decoding. If $\mathbf{y} = \mathbf{c} + \mathbf{e}$ is received with error vector $\mathbf{e}$ of weight ≤ 1, compute the *syndrome*

$$\mathbf{s} = \mathbf{H}\mathbf{y} = \mathbf{H}(\mathbf{c} + \mathbf{e}) = \mathbf{H}\mathbf{e} \in \{0,1\}^r.$$

If $\mathbf{s} = \mathbf{0}$: no error occurred. If $\mathbf{s} \neq \mathbf{0}$: $\mathbf{s}$ equals the column of $\mathbf{H}$ indexed by the error position (since $\mathbf{H}\mathbf{e} = \mathbf{H}e_i$ = column i of $\mathbf{H}$ = binary representation of i), so flip bit i, where i is the integer whose binary representation is $\mathbf{s}$.

Hamming codes are perfect. The balls of radius 1 around the 2^k codewords are disjoint (since $d_{\min} = 3 > 2$) and each has size $V(n,1) = 1 + n = 1 + (2^r - 1) = 2^r$. Their union has size $2^k \cdot 2^r = 2^{k+r} = 2^n$. So every binary string of length $n = 2^r - 1$ lies in exactly one ball, confirming perfection.

Example — The $[7,4,3]$ Hamming Code

Take $r = 3$: $n = 7$, $k = 4$, $d = 3$. The parity-check matrix is:

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

(columns are binary representations of $1, 2, \ldots, 7$). The $2^4 = 16$ codewords include 0000000 and 1010101 (check: $\mathbf{H} \cdot (1,0,1,0,1,0,1)^\top = (0+0+1, 0+1+0, 1+0+1)^\top = (1,1,0)^\top \bmod 2$ — wait, let us reorder columns so column j is the binary representation of j: column $1 = (0,0,1)^\top$, column $2 = (0,1,0)^\top$, column $3 = (0,1,1)^\top$, column $4 = (1,0,0)^\top$, column $5 = (1,0,1)^\top$, column $6 = (1,1,0)^\top$, column $7 = (1,1,1)^\top$.) Encoding: The information bits $u_1u_2u_3u_4$ are placed in positions $3, 5, 6, 7$ (non-power-of-2 positions). The parity bits p_1, p_2, p_3 in positions $1, 2, 4$ are chosen so that $\mathbf{H}\mathbf{c} = \mathbf{0}$. For $u_1u_2u_3u_4 = 1011$:

$$\begin{aligned} p_1 &= u_1 \oplus u_2 \oplus u_4 = 1 \oplus 0 \oplus 1 = 0 \\ p_2 &= u_1 \oplus u_3 \oplus u_4 = 1 \oplus 1 \oplus 1 = 1 \\ p_3 &= u_2 \oplus u_3 \oplus u_4 = 0 \oplus 1 \oplus 1 = 0 \end{aligned}$$

So the codeword is $c = (p_1, p_2, u_1, p_3, u_2, u_3, u_4) = (0,1,1,0,0,1,1) = 0110011$. Decoding: Suppose $\mathbf{y} = 0110111$ (bit position 5 flipped). Syndrome: $\mathbf{s} = \mathbf{H}\mathbf{y}$, computing:

$$\begin{aligned} s_1 &= y_4 \oplus y_5 \oplus y_6 \oplus y_7 = 0 \oplus 1 \oplus 1 \oplus 1 = 1 \\ s_2 &= y_2 \oplus y_3 \oplus y_6 \oplus y_7 = 1 \oplus 1 \oplus 1 \oplus 1 = 0 \\ s_3 &= y_1 \oplus y_3 \oplus y_5 \oplus y_7 = 0 \oplus 1 \oplus 1 \oplus 1 = 1 \end{aligned}$$

Syndrome $\mathbf{s} = (1,0,1)^\top$ in binary is 5. Flip bit 5: $0110\mathbf{1}11 \to 0110011$. Correct!

1.4 The Gilbert-Varshamov Bound and Probabilistic Existence Arguments

Upper bounds tell us what is impossible. Lower bounds tell us that good codes *exist*, often without constructing them. The most important existence lower bound is due to Gilbert (1952) and independently Varshamov (1957).

Gilbert-Varshamov (GV) Bound

For any n and d with $d \leq n$, there exists a binary (n, k, d) code with

$$k \geq n - \log_2 V(n, d-1) = n - \log_2 \sum_{j=0}^{d-1} \binom{n}{j}.$$

In terms of rate $R = k/n$ and relative distance $\delta = d/n$, the asymptotic form is:

$$R \geq 1 - h_b(\delta), \qquad 0 \leq \delta \leq 1/2.$$

Proof (greedy construction). We build the code $\mathcal{C}$ greedily. Start with $\mathcal{C} = \{\mathbf{0}\}$. At each step, add a new codeword $\mathbf{c}$ that is at Hamming distance $\geq d$ from all codewords already in $\mathcal{C}$, as long as such a $\mathbf{c}$ exists.

The process terminates when every $\mathbf{x} \in \{0,1\}^n$ is within distance $d-1$ of some codeword in $\mathcal{C}$ (otherwise we could add $\mathbf{x}$). At termination, the balls of radius $d-1$ around codewords cover all of $\{0,1\}^n$:

$$\bigcup_{\mathbf{c} \in \mathcal{C}} B(\mathbf{c}, d-1) = \{0,1\}^n.$$

Since $|B(\mathbf{c}, d-1)| = V(n, d-1)$ for every $\mathbf{c}$, a union bound gives $|\mathcal{C}| \cdot V(n, d-1) \geq 2^n$, i.e. $|\mathcal{C}| \geq 2^n / V(n, d-1)$. The final code has $k \geq n - \log_2 V(n, d-1)$. Moreover, by construction, any two distinct codewords are at distance $\geq d$, so $d_{\min}(\mathcal{C}) \geq d$. ■

Comparison with the Hamming bound. The Hamming bound says $R \leq 1 - h_b(\delta/2)$, while the GV bound says $R \geq 1 - h_b(\delta)$. Since $h_b(\delta) > h_b(\delta/2)$ for $\delta > 0$ (the binary entropy function is concave), there is a gap between the two bounds, especially for large δ. Whether this gap is tight — i.e. whether there exist codes beating the GV bound — is one of the deepest open problems in combinatorics and coding theory.

The GV gap problem. For $\delta \in (0, 1/2)$, the Hamming bound gives $R \leq 1 - h_b(\delta/2)$ and the GV bound gives $R \geq 1 - h_b(\delta)$. For any $\delta > 0$, it is known that $1 - h_b(\delta) < 1 - h_b(\delta/2)$, so the bounds are incomparable and there is a gap. The correct answer for $A_{\text{lin}}(n, d)$ (for linear codes) is not known. For q-ary codes with $q \to \infty$, algebraic-geometry codes (Goppa codes) beat GV, but for $q = 2$ the question remains open as of 2025.

1.5 Weight Enumerators and the MacWilliams Identity

A code's combinatorial structure is encoded in its *weight enumerator polynomial*.

Weight Enumerator

For a binary code $\mathcal{C}$ with codewords $\mathbf{c}_1, \ldots, \mathbf{c}_M$, the *weight enumerator* is the polynomial

$$W_{\mathcal{C}}(x, y) = \sum_{i=1}^{M} x^{n - w_H(\mathbf{c}_i)} y^{w_H(\mathbf{c}_i)} = \sum_{j=0}^{n} A_j \, x^{n-j} y^j,$$

where $A_j = |\{\mathbf{c} \in \mathcal{C} : w_H(\mathbf{c}) = j\}|$ is the number of codewords of weight j.

The weight enumerator captures full information about the weight distribution: $A_0 = 1$ (assuming $\mathbf{0} \in \mathcal{C}$), $A_1 = A_2 = \ldots = A_{d-1} = 0$, and $\sum_j A_j = M$. For example, the weight

enumerator of the $[7, 4, 3]$ Hamming code is

$$W(x, y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7,$$

reflecting that among the 16 codewords, there is 1 of weight 0, 7 of weight 3, 7 of weight 4, and 1 of weight 7.

The most powerful result in the theory of weight enumerators is the MacWilliams identity, which relates the weight enumerator of a linear code to that of its dual.

MacWilliams Identity

If $\mathcal{C}$ is a binary linear $[n, k]$ code with dual code $\mathcal{C}^\perp$ (an $[n, n-k]$ code), then

$$W_{\mathcal{C}^\perp}(x, y) = \frac{1}{|\mathcal{C}|} W_{\mathcal{C}}(x + y, x - y).$$

Proof sketch. The key identity is the Fourier-analytic computation:

$$\sum_{\mathbf{c} \in \mathcal{C}} (-1)^{\mathbf{u} \cdot \mathbf{c}} = \begin{cases} |\mathcal{C}| & \text{if } \mathbf{u} \in \mathcal{C}^\perp \\ 0 & \text{otherwise,} \end{cases}$$

which follows from the character theory of $(\mathbb{Z}/2\mathbb{Z})^n$. Using this, one computes

$$\begin{aligned} \sum_{\mathbf{v} \in \mathcal{C}^\perp} x^{n - w_H(\mathbf{v})} y^{w_H(\mathbf{v})} &= \sum_{\mathbf{v} \in \mathcal{C}^\perp} \prod_{i=1}^{n} (x \cdot \mathbf{1}\{v_i = 0\} + y \cdot \mathbf{1}\{v_i = 1\}) \\ &= \frac{1}{|\mathcal{C}|} \sum_{\mathbf{u} \in \{0,1\}^n} W_{\mathcal{C}}(x + y, x - y) \cdot (\text{algebra omitted}). \end{aligned}$$

The full computation uses the substitution $(-1)^{v_i} = (1 - 2v_i)$ and interchange of sums; the complete proof is given in Appendix A. $\blacksquare$

The MacWilliams identity has far-reaching consequences: it constrains the weight distribution of a code given knowledge of its dual, and is the foundation of linear programming bounds (Delsarte's method) which give the best known upper bounds on $A(n, d)$ for many parameter regimes.

Part 2 — Linear Codes and Finite Fields

Finite fields $\mathbb{F}_q$: construction and arithmetic. Linear codes: generator matrices, parity-check matrices, systematic form, dual codes. Cyclic codes: polynomial representation, generator polynomial, BCH codes. Reed-Solomon codes as MDS codes: construction over $\mathbb{F}_q$, minimum distance proof. Berlekamp-Massey decoding algorithm. Euclidean algorithm decoder.

2.1 Finite Fields: Construction and Arithmetic

Linear codes are defined over *finite fields* (also called *Galois fields*). Before developing the algebra of linear codes, we must understand the arithmetic of these fields.

Finite Field (Galois Field)

A *finite field* $\mathbb{F}_q$ is a finite set of q elements equipped with addition and multiplication operations satisfying all field axioms (commutativity, associativity, distributivity, existence of additive and multiplicative identities, and existence of inverses for all nonzero elements).
Finite fields exist if and only if $q = p^m$ for a prime p and integer $m \geq 1$. For each prime power $q = p^m$, the finite field $\mathbb{F}_q$ is unique up to isomorphism.

The prime field $\mathbb{F}_p$. The simplest finite field is $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z} = \{0, 1, \dots, p-1\}$ with arithmetic modulo p. For example, $\mathbb{F}_2 = \{0, 1\}$ with $1 + 1 = 0$. Every finite field contains a subfield isomorphic to $\mathbb{F}_p$ for some prime p (the *characteristic* of the field).

Extension fields $\mathbb{F}_{p^m}$. To construct $\mathbb{F}_{p^m}$ for $m \geq 2$, choose an *irreducible polynomial* $f(x) \in \mathbb{F}_p[x]$ of degree m (i.e. f has no roots in $\mathbb{F}_p$ and does not factor over $\mathbb{F}_p$). Then define

$$\mathbb{F}_{p^m} = \mathbb{F}_p[x]/(f(x)) = \{a_0 + a_1 x + \cdots + a_{m-1}x^{m-1} : a_i \in \mathbb{F}_p\},$$

the set of polynomials over $\mathbb{F}_p$ of degree less than m, with multiplication performed modulo $f(x)$. This set has p^m elements, and is a field because f is irreducible.

Example — Construction of $\mathbb{F}_4$

Take $p = 2$, $m = 2$. The polynomial $f(x) = x^2 + x + 1$ is irreducible over $\mathbb{F}_2$ (it has no roots: $f(0) = 1 \neq 0$, $f(1) = 1 + 1 + 1 = 1 \neq 0$).
$\mathbb{F}_4 = \mathbb{F}_2[x]/(x^2 + x + 1) = \{0, 1, \alpha, \alpha + 1\}$, where α is the image of x and satisfies $\alpha^2 + \alpha + 1 = 0$, i.e. $\alpha^2 = \alpha + 1$.
Multiplication table (using $\alpha^2 = \alpha+1$, $\alpha^3 = \alpha \cdot \alpha^2 = \alpha(\alpha+1) = \alpha^2+\alpha = (\alpha+1)+\alpha = 1$, so α has order 3):

$\cdot$	0	1	α	$\alpha+1$
0	0	0	0	0
1	0	1	α	$\alpha+1$
α	0	α	$\alpha+1$	1
$\alpha+1$	0	$\alpha+1$	1	α

Note $(\alpha + 1)^{-1} = \alpha$ (since $(\alpha + 1) \cdot \alpha = 1$).

The multiplicative group $\mathbb{F}_q^*$ is cyclic. The set of nonzero elements $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$ forms a multiplicative group of order $q-1$. A fundamental theorem of algebra states that this group is cyclic: there exists an element $\alpha \in \mathbb{F}_q^*$, called a *primitive element* (or *generator*), such that $\{1, \alpha, \alpha^2, \dots, \alpha^{q-2}\} = \mathbb{F}_q^*$. Every nonzero element can be written as a power of α, making exponent arithmetic in $\mathbb{F}_q^*$ equivalent to integer arithmetic modulo $q - 1$.

2.2 Linear Codes: Generator and Parity-Check Matrices

Linear Code

A *binary linear code* of length n and dimension k (written $[n,k]$ code) is a k-dimensional subspace $\mathcal{C}$ of $\mathbb{F}_2^n$. More generally, a linear code over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$.

The generator matrix G. A k-dimensional subspace of $\mathbb{F}_q^n$ is the row space of a $k \times n$ matrix $\mathbf{G}$ of rank k:

$$\mathcal{C} = \{\mathbf{uG} : \mathbf{u} \in \mathbb{F}_q^k\}.$$

Encoding maps information vector $\mathbf{u} \in \mathbb{F}_q^k$ to codeword $\mathbf{c} = \mathbf{uG}$.

The parity-check matrix H. The *dual code* $\mathcal{C}^\perp$ is the orthogonal complement of $\mathcal{C}$:

$$\mathcal{C}^\perp = \{\mathbf{v} \in \mathbb{F}_q^n : \mathbf{v} \cdot \mathbf{c} = 0 \text{ for all } \mathbf{c} \in \mathcal{C}\}.$$

$\mathcal{C}^\perp$ is itself a linear code of dimension $n-k$. Its generator matrix $\mathbf{H}$ (of size $(n-k) \times n$) serves as the *parity-check matrix* of $\mathcal{C}$:

$$\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{Hc}^\top = \mathbf{0}\}.$$

The relation $\mathbf{GH}^\top = \mathbf{0}$ holds (rows of $\mathbf{G}$ are in $\mathcal{C}$, rows of $\mathbf{H}$ are in $\mathcal{C}^\perp$).

Systematic form. By row-reducing $\mathbf{G}$, we can put it in *systematic* (standard) form $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$, where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n-k)$ matrix. The corresponding parity-check matrix is $\mathbf{H} = [-\mathbf{P}^\top \mid \mathbf{I}_{n-k}]$ (or $[\mathbf{P}^\top \mid \mathbf{I}_{n-k}]$ over $\mathbb{F}_2$ since $-1 = 1$). Encoding is then explicit: $\mathbf{c} = (\mathbf{u}, \mathbf{uP})$, where $\mathbf{u}$ are the k information bits and $\mathbf{uP}$ are the $n-k$ parity bits.

Minimum distance of a linear code. For a linear code, $d_{\min}(\mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}, \mathbf{c} \neq \mathbf{0}} w_H(\mathbf{c})$, since the difference of two codewords is again a codeword (by linearity) and $d_H(\mathbf{c}, \mathbf{c}') = w_H(\mathbf{c} - \mathbf{c}')$. Thus the minimum distance equals the minimum weight of any nonzero codeword.

Connection to parity-check matrix. A linear code has minimum distance d if and only if every $d-1$ columns of $\mathbf{H}$ are linearly independent (over $\mathbb{F}_q$), and some d columns are linearly dependent. Proof: $w_H(\mathbf{c}) = j$ iff $\mathbf{Hc}^\top = \mathbf{0}$ with $\mathbf{c}$ supported on j positions iff the j columns of $\mathbf{H}$ corresponding to the support of $\mathbf{c}$ are linearly dependent. Thus the minimum weight equals the minimum number of linearly dependent columns of $\mathbf{H}$.

Example — Systematic encoding of a $[6,3]$ code

Suppose

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = [\mathbf{I}_3 \mid \mathbf{P}], \quad \mathbf{P} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Information word $\mathbf{u} = (1,0,1)$: codeword $\mathbf{c} = \mathbf{uG} = (1,0,1) + (0,0,0) + (0,0,1,0,1,1)$. More carefully: $\mathbf{c} = 1 \cdot \text{row}_1 + 0 \cdot \text{row}_2 + 1 \cdot \text{row}_3 = (1,0,0,1,1,0) + (0,0,1,0,1,1) = (1,0,1,1,0,1)$.

Parity-check matrix: $\mathbf{H} = [\mathbf{P}^\top \mid \mathbf{I}_3] = \left(\begin{smallmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{smallmatrix}\right)$. Verification: $\mathbf{Hc}^\top = (1+1+1, 1+1+1, 0+1+1)^\top = (1,1,0)^\top$ over $\mathbb{F}_2$ — this should be $\mathbf{0}$; let us recheck. $c = (1,0,1,1,0,1)$. Row 1 of $\mathbf{H}$: $1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 = 1 + 0 + 0 + 1 + 0 + 0 = 0$. Row 2: $1 + 0 + 1 + 0 + 0 + 0 = 0$. Row 3: $0 + 0 + 1 + 0 + 0 + 1 = 0$. ✓

2.3 Cyclic Codes and the Polynomial Representation

A particularly structured and computationally efficient family of linear codes are the *cyclic codes*. These admit a polynomial representation that makes encoding and decoding reducible to polynomial arithmetic.

Cyclic Code

A linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is *cyclic* if for every codeword $(c_0, c_1, \dots, c_{n-1}) \in \mathcal{C}$, the cyclic shift $(c_{n-1}, c_0, c_1, \dots, c_{n-2})$ is also a codeword.
Equivalently, identify codeword $(c_0, \dots, c_{n-1})$ with the polynomial $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in \mathbb{F}_q[x]/(x^n - 1)$. Then $\mathcal{C}$ is cyclic iff $\mathcal{C}$ is an ideal in the ring $\mathbb{F}_q[x]/(x^n - 1)$.

Since $\mathbb{F}_q[x]$ is a principal ideal domain, every ideal in $\mathbb{F}_q[x]/(x^n - 1)$ is generated by a single polynomial $g(x)$, called the *generator polynomial* of $\mathcal{C}$:

$$\mathcal{C} = \{u(x)g(x) \bmod (x^n - 1) : u(x) \in \mathbb{F}_q[x], \deg u < k\},$$

where $g(x)$ divides $x^n - 1$ and $k = n - \deg g$.

The parity-check polynomial. Write $x^n - 1 = g(x)h(x)$. Then $h(x)$ is the *parity-check polynomial*: a polynomial $c(x)$ is a codeword iff $c(x)h(x) \equiv 0 \pmod{x^n - 1}$.

Encoding. Given information polynomial $u(x)$ of degree $< k$, encode as $c(x) = u(x)g(x)$ (non-systematic) or, in systematic form, as $c(x) = x^{n-k}u(x) - r(x)$, where $r(x) = x^{n-k}u(x) \bmod g(x)$ is the remainder upon dividing $x^{n-k}u(x)$ by $g(x)$.

2.4 Reed-Solomon Codes: Construction and Minimum Distance

Reed-Solomon (RS) codes are the most important family of MDS codes, achieving the Singleton bound $d = n - k + 1$. They are used in virtually every error-correction system: CDs, DVDs, Blu-ray discs, QR codes, RAID storage, and deep-space communication.

Reed-Solomon Code

Let $\mathbb{F}_q$ be a finite field with q elements, and let $\alpha_1, \dots, \alpha_n$ be n distinct elements of $\mathbb{F}_q$ (so $n \le q$). The *Reed-Solomon code* $\mathrm{RS}(n, k)$ over $\mathbb{F}_q$ with evaluation points $\alpha_1, \dots, \alpha_n$ is

$$\mathrm{RS}(n, k) = \{(f(\alpha_1), f(\alpha_2), \dots, f(\alpha_n)) : f \in \mathbb{F}_q[x], \deg f \le k - 1\}.$$

This is an $[n, k, d]$ code over $\mathbb{F}_q$ (with q-ary alphabet).

The minimum distance of RS codes is $d = n - k + 1$. This follows immediately from the Singleton bound, once we verify that $\mathrm{RS}(n, k)$ is a linear code of dimension k:

- *Linearity*: the map $f \mapsto (f(\alpha_1), \dots, f(\alpha_n))$ is linear in f, so $\mathrm{RS}(n, k)$ is a linear code.
- *Dimension k*: the evaluation map is injective on polynomials of degree $\le k - 1$ (a nonzero polynomial of degree $\le k-1$ can have at most $k-1 < n$ roots, so its evaluation on n distinct points cannot be the all-zero vector). The space of polynomials of degree $\le k - 1$ has dimension k, so the image has dimension k.
- *Minimum distance $n-k+1$*: A nonzero polynomial of degree $\le k-1$ has at most $k-1$ roots among $\alpha_1, \dots, \alpha_n$, so its evaluation vector has at most $k - 1$ zeros, hence weight at least $n - (k - 1) = n - k + 1$. The Singleton bound then forces equality.

Generator matrix of RS codes. The generator matrix of RS(n, k) is the *Vandermonde matrix*:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \alpha_3^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix} \in \mathbb{F}_q^{k \times n}.$$

Encoding: $\mathbf{c} = \mathbf{uG}$, where $\mathbf{u} = (u_0, \ldots, u_{k-1})$ corresponds to the polynomial $f(x) = u_0 + u_1 x + \cdots + u_{k-1} x^{k-1}$.

Example — RS$(7, 3)$ code over $\mathbb{F}_8$

Take $\mathbb{F}_8 = \mathbb{F}_{2^3}$, generated by a root α of the irreducible polynomial $x^3 + x + 1$ over $\mathbb{F}_2$. The elements of $\mathbb{F}_8^*$ are $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$ with $\alpha^7 = 1$. The RS$(7, 3)$ code uses all 7 nonzero elements as evaluation points. It has parameters $[7, 3, 5]$ over $\mathbb{F}_8$: rate $3/7 \approx 0.43$, correcting up to $\lfloor 4/2 \rfloor = 2$ errors.
For information polynomial $f(x) = 1 + \alpha x + \alpha^2 x^2$, the codeword is $(f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^6))$. Computing $f(1) = 1 + \alpha + \alpha^2$; $f(\alpha) = 1 + \alpha^2 + \alpha^4$; and so on using the arithmetic of $\mathbb{F}_8$.

2.5 The Berlekamp-Massey Algorithm

RS codes can correct up to $t = \lfloor (d-1)/2 \rfloor = \lfloor (n-k)/2 \rfloor$ errors. The challenge is *efficient decoding*. The Berlekamp-Massey algorithm finds the shortest linear recurrence satisfied by the syndrome sequence, which is equivalent to finding the *error-locator polynomial*. This polynomial has roots at the reciprocals of the error locations, enabling recovery of the error positions and magnitudes.

Setup. Suppose codeword $\mathbf{c}$ is sent and $\mathbf{y} = \mathbf{c} + \mathbf{e}$ is received, with error vector $\mathbf{e}$ having weight $w \leq t$. Let $E = \{i : e_i \neq 0\}$ be the set of error positions and e_i the error magnitudes. Define the *syndromes*

$$S_j = \sum_{i=1}^{n} y_i \alpha_i^j = \sum_{i=1}^{n} e_i \alpha_i^j, \quad j = 0, 1, \ldots, 2t-1,$$

where the second equality uses $c_i =$ a codeword value which satisfies $\sum_i c_i \alpha_i^j = 0$ for $j = 0, \ldots, d-2$.

The error-locator polynomial. Define $\Lambda(z) = \prod_{i \in E} (1 - \alpha_i z) = 1 + \Lambda_1 z + \cdots + \Lambda_w z^w$. The roots of Λ are $\{alpha_i^{-1} : i \in E\}$, the reciprocals of the error-locating elements. The key algebraic identity is Newton's identity:

$$S_j + \Lambda_1 S_{j-1} + \Lambda_2 S_{j-2} + \cdots + \Lambda_w S_{j-w} = 0, \quad j = w, w+1, \ldots, 2t-1.$$

This says that the syndrome sequence $(S_0, S_1, \ldots)$ satisfies a linear recurrence with characteristic polynomial $\Lambda(z)$.

The Berlekamp-Massey algorithm finds the shortest such recurrence — i.e., the minimum-degree Λ — from the sequence $S_0, \ldots, S_{2t-1}$.

Berlekamp-Massey Algorithm

Input: Sequence $S_0, S_1, \dots, S_{2t-1}$.
Output: The shortest LFSR (linear feedback shift register) with characteristic polynomial $\Lambda(z)$ generating the sequence.
Initialize: $\Lambda(z) \leftarrow 1$, $B(z) \leftarrow 1$, $L \leftarrow 0$, $b \leftarrow 1$, $m \leftarrow 1$.
For $n = 0, 1, \dots, 2t-1$:

1. Compute discrepancy: $\delta \leftarrow S_n + \sum_{j=1}^{L} \Lambda_j S_{n-j}$.
2. If $\delta = 0$: set $m \leftarrow m + 1$.
3. If $\delta \neq 0$ and $2L \leq n$: $\Lambda(z) \leftarrow \Lambda(z) - (\delta/b) z^m B(z)$, $B(z) \leftarrow \Lambda_{\text{old}}(z)$, $L \leftarrow n+1-L$, $b \leftarrow \delta$, $m \leftarrow 1$.
4. If $\delta \neq 0$ and $2L > n$: $\Lambda(z) \leftarrow \Lambda(z) - (\delta/b) z^m B(z)$, $m \leftarrow m + 1$.

Complexity. The Berlekamp-Massey algorithm runs in $O(t^2)$ field operations, far faster than naive Gaussian elimination on the $t \times t$ syndrome matrix (which is also $O(t^3)$ but with larger constants). The algorithm is optimal in the sense that it processes each syndrome element exactly once and produces the unique minimum-degree solution.

Finding error values. Once $\Lambda(z)$ is found, the error positions are $\{i : \Lambda(\alpha_i^{-1}) = 0\}$, found by evaluating Λ at $\alpha^{-1}, \alpha^{-2}, \dots, \alpha^{-n}$ (Chien search). The error magnitudes are then found by solving the Forney formula.

2.6 BCH Codes and Their Distance Properties

Bose-Chaudhuri-Hocquenghem (BCH) codes generalize Hamming codes and constitute the natural binary subfamily of RS codes. They are cyclic codes defined by specifying the roots of the generator polynomial.

BCH Code

Let α be a primitive n-th root of unity in an extension of $\mathbb{F}_q$. A *BCH code of designed distance* δ is the cyclic code of length n over $\mathbb{F}_q$ with generator polynomial

$$g(x) = \operatorname{lcm}(m_b(x), m_{b+1}(x), \dots, m_{b+\delta-2}(x)),$$

where $m_j(x)$ is the minimal polynomial of α^j over $\mathbb{F}_q$, for some $b \geq 1$.

BCH bound. If a cyclic code has generator polynomial with $\delta-1$ consecutive roots $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+\delta-2}$, then $d_{\min} \geq \delta$.

Proof. Suppose $\mathbf{c}$ is a codeword of weight $w < \delta$. Then $c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$ has $c(\alpha^j) = 0$ for $j = b, \dots, b+\delta-2$ (since $g(x) \mid c(x)$). The matrix

$$\begin{pmatrix} \alpha^{bi_1} & \alpha^{bi_2} & \cdots & \alpha^{bi_w} \\ \alpha^{(b+1)i_1} & \alpha^{(b+1)i_2} & \cdots & \alpha^{(b+1)i_w} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{(b+\delta-2)i_1} & \cdots & \cdots & \alpha^{(b+\delta-2)i_w} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_w} \end{pmatrix} = \mathbf{0},$$

where $i_1, \dots, i_w$ are the support positions of $\mathbf{c}$, must hold. If $w < \delta$, this is an overdetermined homogeneous system; its coefficient matrix is a Vandermonde-like matrix of rank $\min(w, \delta-1) = w$, so the only solution is $c_{i_j} = 0$ for all j, contradicting $\mathbf{c} \neq \mathbf{0}$. Hence $w \geq \delta$. ■

Part 3 — Modern Iterative Codes

Sparse graph codes: LDPC codes via Tanner graphs. Variable and check node perspective. The sum-product (belief propagation) algorithm in detail. Message passing on trees: exact computation and the cycle problem. Density evolution for the BEC and the BSC. Capacity-approaching LDPC code sequences. Turbo codes: parallel concatenated convolutional codes, the BCJR algorithm for soft decoding, the turbo principle and iterative decoding.

3.1 Low-Density Parity-Check Codes

The algebraic codes of Part 2 — Hamming, Reed-Solomon, BCH — have structured and efficient decoding algorithms, but their minimum distances are modest when measured against the Shannon limit, and their encoding complexity can be high. The breakthrough of modern coding theory was the discovery of *iterative codes*: codes with sparse graphical representations that can be decoded efficiently via message-passing algorithms, approaching the Shannon capacity for sufficiently large block lengths.

Low-density parity-check (LDPC) codes were introduced by Gallager in 1963 but largely forgotten for three decades, until their rediscovery by MacKay and Neal in 1996. They achieve rates within fractions of a decibel of the Shannon limit.

LDPC Code and Tanner Graph

An $[n, k]$ LDPC code is a linear code whose parity-check matrix $\mathbf{H} \in \{0,1\}^{m \times n}$ (with $m = n - k$) is *sparse*: the number of 1s per row and per column is small (much less than m or n, respectively).
The *Tanner graph* of $\mathbf{H}$ is a bipartite graph $G = (V \cup C, E)$, where:

- $V = \{v_1, \dots, v_n\}$ are *variable nodes*, one per code bit.
- $C = \{c_1, \dots, c_m\}$ are *check nodes*, one per parity-check equation.
- Edge $(v_j, c_i) \in E$ iff $H_{ij} = 1$ (variable j participates in check i).

Regular and irregular LDPC codes. A (d_v, d_c)-*regular* LDPC code has Tanner graph in which every variable node has degree d_v and every check node has degree d_c. The rate satisfies $R = 1 - m/n = 1 - d_v/d_c$ (by counting edges: $nd_v = md_c$).

Irregular LDPC codes have variable nodes and check nodes with varying degrees, described by *degree distribution polynomials*

$$\lambda(x) = \sum_i \lambda_i x^{i-1}, \qquad \rho(x) = \sum_i \rho_i x^{i-1},$$

where λ_i (resp. ρ_i) is the fraction of edges adjacent to variable (resp. check) nodes of degree i. The design rate of the code is $R = 1 - \int_0^1 \rho(x)dx / \int_0^1 \lambda(x)dx$.

3.2 The Sum-Product Algorithm (Belief Propagation)

The sum-product algorithm, also known as the *belief propagation* (BP) algorithm, is the inference engine for LDPC decoding. It is an instance of variable elimination / dynamic programming on the Tanner graph, computing (exactly on trees, approximately on graphs with cycles) the *a posteriori probability* (APP) of each bit given the received channel output.

Message-passing on the Tanner graph. At each iteration, messages are passed along edges of the Tanner graph in both directions: from variable nodes to check nodes, and from check nodes to variable nodes.

Let $\mu^{(\ell)}_{v_j \to c_i}$ denote the message from variable node v_j to check node c_i at iteration ℓ, and $\mu^{(\ell)}_{c_i \to v_j}$ the reverse. We work in the *log-likelihood ratio* (*LLR*) domain: messages represent

$$L^{(\ell)}_{v_j \to c_i} = \log \frac{P(\text{bit}_j = 0 \mid y_j, \text{messages from } \mathcal{N}(v_j) \setminus c_i)}{P(\text{bit}_j = 1 \mid y_j, \text{messages from } \mathcal{N}(v_j) \setminus c_i)},$$

where $\mathcal{N}(v_j)$ is the set of check nodes adjacent to v_j.

Sum-Product Update Rules (LLR Domain)

Let $L^{\text{ch}}_j = \log \frac{p(y_j|\text{bit}_j=0)}{p(y_j|\text{bit}_j=1)}$ be the channel LLR for bit j.
Variable-to-check messages (iteration $\ell \geq 1$):

$$L^{(\ell)}_{v_j \to c_i} = L^{\text{ch}}_j + \sum_{c_{i'} \in \mathcal{N}(v_j) \setminus c_i} L^{(\ell-1)}_{c_{i'} \to v_j}.$$

Check-to-variable messages:

$$L^{(\ell)}_{c_i \to v_j} = 2 \operatorname{arctanh}\left(\prod_{v_{j'} \in \mathcal{N}(c_i) \setminus v_j} \tanh\left(\frac{L^{(\ell)}_{v_{j'} \to c_i}}{2} \right) \right).$$

Bit APP estimate (after ℓ iterations):

$$L^{(\ell)}_j = L^{\text{ch}}_j + \sum_{c_i \in \mathcal{N}(v_j)} L^{(\ell)}_{c_i \to v_j}.$$

Decode bit j as $\hat{b}_j = 0$ if $L^{(\ell)}_j > 0$, else $\hat{b}_j = 1$.

Why does this work? On a tree-structured Tanner graph (no cycles), the sum-product algorithm computes exact marginal posteriors in $O(n \cdot d_{\max})$ operations after a finite number of iterations equal to half the girth. This follows from the exact marginalization interpretation: each message $L^{(\ell)}_{v_j \to c_i}$ encodes the log-likelihood ratio of bit j, given all channel observations in the subtree rooted at v_j away from c_i, up to depth 2ℓ.

On graphs with cycles (as in practice), the algorithm is an approximation. Its empirical performance is excellent for long codes with large girth, and it provably achieves capacity for carefully designed irregular LDPC sequences (as shown by density evolution in Section 3.3).

The check-node update via the "box-plus" operation. A numerically stable alternative to the tanh formula above uses the *box-plus* operation $\boxplus$ defined in the min-sum domain. For a single parity-check node of degree d_c with incoming LLRs $L_1, \ldots, L_{d_c}$:

$$L_{\text{out}} = \left(\prod_{j=1}^{d_c - 1} \operatorname{sign}(L_j) \right) \cdot f\left(\sum_{j=1}^{d_c - 1} f(|L_j|) \right),$$

where $f(x) = \log \coth(x/2) = \log \frac{e^x+1}{e^x-1}$ is the Gallagher f-function. Note $f = f^{-1}$ (the function is its own inverse). The *min-sum* approximation replaces $f(\sum_j f(|L_j|))$ with $\min_j |L_j|$, incurring a small performance loss but greatly reducing computational complexity.

Example — BP decoding of a $(3,6)$-regular LDPC code on the BEC

On the binary erasure channel (BEC) with erasure probability ϵ, every received bit is either known perfectly or is an erasure (denoted ?). Messages are simply one of $\{0, 1, ?\}$.
Variable-to-check: send the known value if known, else send ?. *Check-to-variable*: send $0 \oplus \bigoplus_j b_j$ if all other incoming messages are known bits b_j, else send ?.
On a $(3,6)$-regular LDPC code, iterative decoding recovers all erasures if $\epsilon < \epsilon^* \approx 0.429$, where the threshold ϵ^* is the solution to $1 - e^{-3(1-\epsilon)\lambda} = \epsilon$ with the appropriate degree polynomials. The Shannon limit for the BEC is $\epsilon < 1 - R = 1 - 1/2 = 0.5$, so the BP threshold is about 14% below capacity.

3.3 Density Evolution and Capacity-Achieving LDPC Sequences

Density evolution (DE) is a technique for tracking the distribution of BP messages over iterations for an ensemble of random LDPC codes in the large block-length limit. It reduces the analysis of a complex iterative decoder on a random graph to a deterministic recursion on scalar (or functional) quantities.

Density Evolution on the BEC

On the BEC, messages are either "known" (probability $1 - p$) or "erased" (probability p). Let p_ℓ be the probability that a variable-to-check message is an erasure at iteration ℓ. Then:

Initialization: $p_0 = \epsilon$ (the channel erasure probability).

Variable-to-check update: A variable-to-check message is erased iff the channel bit is erased AND all incoming check-to-variable messages are erased. With degree distributions $\lambda(x)$ and $\rho(x)$, the check-to-variable message is erased with probability x_ℓ, and the update is

$$p_{\ell+1} = \epsilon \cdot \lambda(1 - \rho(1 - p_\ell)).$$

Here the factor $\lambda(1-\rho(1-p_\ell))$ accounts for the fraction of edges from variable nodes where all other check-to-variable messages are erased.

The LDPC code ensemble with degree distributions (λ, ρ) achieves *vanishing block error probability* as $n \to \infty$ for channel erasure probability ϵ iff the density evolution fixed-point equation

$$p = \epsilon \cdot \lambda(1 - \rho(1 - p))$$

has $p = 0$ as its only fixed point in $[0, 1)$. The *threshold* $\epsilon^*(\lambda, \rho)$ is the supremum of ϵ for which this holds.

Capacity-achieving LDPC sequences. Luby, Mitzenmacher, Shokrollahi, and Spielman (2001) showed that there exist sequences of irregular LDPC codes with degree distributions optimized so that the threshold ϵ^* approaches the capacity $1 - R$ of the BEC as the rate decreases. More precisely, for any $\delta > 0$, there exist (λ, ρ) such that $\epsilon^* \geq (1 - R) - \delta$. This was the first rigorous demonstration that a practical family of codes approaches capacity on the BEC.

Density Evolution on the AWGN Channel

On the additive white Gaussian noise (AWGN) channel with noise variance σ^2, channel LLRs follow the distribution $L^{\text{ch}} \sim \mathcal{N}(2/\sigma^2, 4/\sigma^2)$ (for BPSK signaling). Density evolution tracks the full density (probability distribution) of BP messages, which evolves as:

Variable-to-check density: $p_{v\to c}^{(\ell)} = p^{\mathrm{ch}} * p_{c\to v}^{(\ell-1)} * \cdots * p_{c\to v}^{(\ell-1)}$ (convolution of $d_v - 1$ incoming densities plus the channel density), where $*$ denotes convolution in LLR space.

Check-to-variable density: The tanh rule makes this more complex. One tracks the densities through the tanh transformation, the product (convolution of log-tanh transforms), and back.

In practice, density evolution on the AWGN channel is implemented numerically by discretizing the LLR distribution to a vector of probabilities and updating the vector at each iteration. The threshold E_b/N_0^* is the smallest signal-to-noise ratio for which the density evolution converges to the all-known (zero-error) distribution.

Protograph LDPC codes. Modern codes (including those used in 5G NR and DVB-S2) are constructed as *protograph lifts*: start from a small *protograph* (a Tanner graph with a few dozen nodes) and create a lifted code of size $n = Z \cdot n_0$ by replacing each protograph edge with a $Z \times Z$ permutation matrix. This enables hardware-efficient encoding and decoding while maintaining the capacity-approaching properties predicted by density evolution on the protograph.

3.4 Turbo Codes and the Turbo Principle

Turbo codes, introduced by Berrou, Glavieux, and Thitimajshima in 1993, caused a revolution in coding theory by achieving within 0.5 dB of the Shannon limit on the AWGN channel at moderately large block lengths. Their key innovation was the *turbo principle*: iterative exchange of *extrinsic information* between two (or more) component decoders.

Parallel Concatenated Convolutional Code (Turbo Code)

A *turbo code* consists of:

- An information sequence $\mathbf{u} = (u_1, \dots, u_k)$.
- A *systematic* output: $\mathbf{u}$ itself is transmitted.
- A first *convolutional code* C_1 encoding $\mathbf{u}$, producing parity bits $\mathbf{v}^{(1)}$.
- An *interleaver* π that permutes $\mathbf{u}$ to produce $\mathbf{u}^{\pi} = \mathbf{u}_{\pi}$.
- A second convolutional code C_2 encoding $\mathbf{u}^{\pi}$, producing parity bits $\mathbf{v}^{(2)}$.

The transmitted codeword is $(\mathbf{u}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)})$ (possibly punctured to increase rate). The overall rate is $1/3$ before puncturing, or higher after.

The BCJR algorithm. Each component decoder must produce *soft* (probabilistic) outputs, not hard bit decisions. The BCJR algorithm (Bahl, Cocke, Jelinek, Raviv, 1974) computes the APP $P(u_t = b \mid \mathbf{y})$ for each information bit u_t, using the trellis of the convolutional code.

A convolutional code is described by a *trellis*: a directed graph in which nodes at time t represent encoder states $s_t \in \mathcal{S}$, and edges from s_t to s_{t+1} correspond to input bit u_t, labeled with the output bits $\mathbf{v}_t$. The BCJR algorithm computes, for each trellis stage t and state s:

BCJR Algorithm

Forward recursion (alpha):

$$\alpha_t(s) = P(s_t = s, y_1^t) = \sum_{s'} \alpha_{t-1}(s')\gamma_t(s', s),$$

where $\gamma_t(s', s) = P(y_t \mid \mathbf{v}_t(s', s)) \cdot P(u_t)$ is the branch metric.

Backward recursion (beta):

$$\beta_t(s) = P(y_{t+1}^T \mid s_t = s) = \sum_{s'} \beta_{t+1}(s')\gamma_{t+1}(s, s').$$

APP computation:

$$P(u_t = b \mid \mathbf{y}) \propto \sum_{(s',s):u_t(s',s)=b} \alpha_{t-1}(s')\gamma_t(s', s)\beta_t(s).$$

Extrinsic LLR:

$$L_e(u_t) = \log \frac{P(u_t = 0 \mid \mathbf{y})}{P(u_t = 1 \mid \mathbf{y})} - L^{\text{ch}}(u_t) - L_a(u_t),$$

where $L_a(u_t)$ is the *a priori* LLR from the other decoder. The extrinsic LLR carries only the information gained from the component code, not the channel or the prior.

Turbo decoding iterations.

1. Decoder 1 runs BCJR on $(y_u, y^{(1)})$ using $L_a^{(1)}(u_t) = 0$ (no a priori). Outputs extrinsic LLRs $L_e^{(1)}(u_t)$.
2. These are passed through the interleaver π to serve as a priori information for Decoder 2.
3. Decoder 2 runs BCJR on $(y_u^\pi, y^{(2)})$ using $L_a^{(2)}(u_{\pi(t)}) = L_e^{(1)}(u_{\pi(t)})$. Outputs extrinsic LLRs $L_e^{(2)}(u_t)$.
4. These are de-interleaved and serve as a priori for Decoder 1 in the next iteration.
5. Repeat for a fixed number of iterations (typically 6–18). At the final iteration, make a hard decision: $\widehat{u}_t = \mathbf{1}\{L_1(u_t) + L_2(u_t) + L^{\text{ch}}(u_t) > 0\}$.

The *turbo principle* is that by repeatedly exchanging only *extrinsic* information (information gained from the current decoder beyond what the other decoder already knows), the decoders cooperate without double-counting evidence. Each iteration refines the soft estimates, and empirically the bit error rate decays rapidly with the number of iterations, approaching the ML performance in the waterfall region.

The interleaver: breaking correlations. The interleaver π is critical to turbo code performance. It ensures that the two component codes see the information bits in different orders, so that the *error events* of the two codes are nearly independent. A good interleaver makes it unlikely that both component codes simultaneously have a low-weight codeword (error pattern), which would cause a catastrophic error even at high SNR. Random interleavers are near-optimal; structured interleavers like the QPP (quadratic permutation polynomial) interleaver used in LTE/5G achieve good performance with efficient hardware implementation.

Part 4 — Data Compression

Prefix-free codes: Kraft inequality, optimality of Huffman coding, redundancy bounds. Arithmetic coding: the interval algorithm, precision and implementation, comparison with Huffman. Universal codes: LZ77, LZ78/LZW, dictionary coder universality theorems. Rate-distortion theory: operational and information-theoretic definitions, the rate-distortion function $R(D)$, Shannon's rate-distortion theorem, Gaussian sources. The Blahut-Arimoto algorithm for $R(D)$. Lossy compression in practice.

4.1 Prefix-Free Codes and the Huffman Algorithm

A *source code* maps symbols from a finite alphabet $\mathcal{X}$ to binary strings. The *expected code length* for a source $X \sim p$ is $\bar{L} = \sum_x p(x)\ell(x)$, where $\ell(x)$ is the length of the codeword for symbol x. Volume I showed that $\bar{L} \geq H(X)$, and $\bar{L} < H(X) + 1$ for the Huffman code. We now prove these bounds rigorously and establish Huffman's optimality.

Prefix-Free Code and the Kraft Inequality

A *prefix-free code* (or *instantaneous code*) assigns codewords such that no codeword is a prefix of another. A binary code with codeword lengths $\ell_1, \ldots, \ell_M$ is prefix-free if and only if the *Kraft inequality* holds:

$$\sum_{i=1}^{M} 2^{-\ell_i} \leq 1.$$

Conversely, for any lengths $\ell_1, \ldots, \ell_M$ satisfying the Kraft inequality, there exists a prefix-free code with those lengths.

Proof of the Kraft inequality. A prefix-free binary code corresponds to a subtree of the infinite binary tree, where each codeword is a leaf. A leaf at depth ℓ corresponds to a binary string of length ℓ, and "blocks" $2^{L-\ell}$ of the leaves at depth L (for any $L \geq \max_i \ell_i$). Prefix-freeness means these blocked sets are disjoint, so their total size is $\sum_i 2^{L-\ell_i} \leq 2^L$, giving $\sum_i 2^{-\ell_i} \leq 1$. Conversely, given lengths satisfying Kraft, one can greedily assign binary strings to leaves. ■

Optimality bound. For any prefix-free code with lengths ℓ_i for symbol x_i with probability p_i:

$$\bar{L} = \sum_i p_i \ell_i \geq H(X).$$

Proof: By Gibbs' inequality (non-negativity of KL divergence):

$$H(X) - \bar{L} = -\sum_i p_i \log_2 p_i - \sum_i p_i \ell_i = \sum_i p_i \log_2 \frac{2^{-\ell_i}}{p_i} \leq \log_2 \sum_i p_i \cdot \frac{2^{-\ell_i}}{p_i} = \log_2 \sum_i 2^{-\ell_i} \leq 0,$$

using Jensen's inequality (concavity of log) and the Kraft inequality. ■

Huffman Code

The *Huffman code* for source distribution $p_1 \geq p_2 \geq \cdots \geq p_M$ is the unique (up to ties) prefix-free code minimizing $\bar{L}$. It is constructed by the following greedy algorithm:

1. Treat each symbol as a singleton tree (leaf) with weight p_i.
2. Merge the two trees with smallest weights into a new tree whose root has weight

equal to the sum of the two children's weights.

3. Assign the bit 0 to the left branch and 1 to the right branch (or vice versa).
4. Repeat until a single tree remains.

The codeword for symbol x_i is the binary string obtained by reading the branch labels on the path from the root to the leaf for x_i.

Huffman codes are optimal.

- The two least probable symbols have the longest codewords and differ only in the last bit (if this were not the case, we could swap a shorter codeword with a longer one to reduce $\bar{L}$, contradicting optimality of the assumed code).
- Merging the two least probable symbols and solving the subproblem gives the optimal code for the merged problem. This optimal substructure justifies the greedy algorithm by induction on M.

Redundancy. The Huffman code achieves $H(X) \leq \bar{L} \leq H(X)+1$. The upper bound comes from choosing $\ell_i = \lceil \log_2(1/p_i) \rceil$, which satisfies Kraft and gives $\ell_i < \log_2(1/p_i) + 1$, so $\bar{L} < H(X) + 1$. The Huffman code is no worse. The redundancy $\bar{L} - H(X)$ approaches 0 as the source probabilities become more skewed (one dominant symbol) or as we code blocks of n symbols (where block Huffman approaches H at rate $1/n$).

Example — Huffman code construction for a 5-symbol source

Source: $p = (0.40, 0.20, 0.15, 0.15, 0.10)$.
Entropy: $H = -0.40\log_2 0.40 - 0.20\log_2 0.20 - 2 \times 0.15\log_2 0.15 - 0.10\log_2 0.10 \approx 0.529 + 0.464 + 2 \times 0.410 + 0.332 = 2.145$ bits.
Huffman construction:

1. Weights: $[0.40, 0.20, 0.15, 0.15, 0.10]$. Merge $0.10 + 0.15 = 0.25$: $[0.40, 0.25, 0.20, 0.15]$.
2. Merge $0.15 + 0.20 = 0.35$: $[0.40, 0.35, 0.25]$.
3. Merge $0.25 + 0.35 = 0.60$: $[0.60, 0.40]$.
4. Merge $0.40 + 0.60 = 1.00$: done.

Assign bits: root gives $\{x_1\} = 0$ and $\{x_2, x_3, x_4, x_5\} = 1$. Continuing: $x_1 \to 0$; $x_2 \to 10$; $\{x_3, x_4, x_5\} \to 11$; $x_3 \to 110$; $\{x_4, x_5\} \to 111$; $x_5 \to 1110$; $x_4 \to 1111$.
Expected length: $0.40 \times 1 + 0.20 \times 2 + 0.15 \times 3 + 0.15 \times 4 + 0.10 \times 4 = 0.40 + 0.40 + 0.45 + 0.60 + 0.40 = 2.25$ bits. Redundancy: $2.25 - 2.145 = 0.105$ bits.

4.2 Arithmetic Coding

Arithmetic coding overcomes the fundamental limitation of Huffman coding — that each symbol must be assigned a whole number of bits — by encoding entire messages with a single real number in $[0, 1)$. The expected code length approaches $H(X)$ without requiring block coding or any structure on the source probability.

Arithmetic Coding: The Basic Algorithm

Given a source with alphabet $\mathcal{X} = \{x_1, \ldots, x_M\}$ and probabilities $(p_1, \ldots, p_M)$ (with cumulative probabilities $F_0 = 0, F_i = \sum_{j \leq i} p_j$), encoding a sequence $(a_1, a_2, \ldots, a_n)$

proceeds as follows:
Initialize: $[\ell, h) \leftarrow [0, 1)$.
For $t = 1, 2, \ldots, n$:

$$\ell \leftarrow \ell + (h - \ell) \cdot F_{a_t - 1}, \qquad h \leftarrow \ell_{\text{old}} + (h_{\text{old}} - \ell_{\text{old}}) \cdot F_{a_t}.$$

(After n steps, the current interval $[\ell, h)$ has length $\prod_{t=1}^{n} p_{a_t}$.)
Output: Any binary string c such that the real number $0.c_1c_2\cdots$ (binary) lies in $[\ell, h)$.

Code length analysis. The interval $[\ell, h)$ after encoding n symbols has length $\prod_t p_{a_t}$. To select a unique point in this interval, we need a binary string of length at most $\lceil -\log_2 \prod_t p_{a_t} \rceil + 1 = \lceil \sum_t \log_2(1/p_{a_t}) \rceil + 1$. The expected code length per symbol satisfies

$$\frac{1}{n}\mathbb{E}[\text{code length}] \leq \frac{1}{n}\mathbb{E}\left[\sum_{t=1}^{n} \log_2 \frac{1}{p_{a_t}}\right] + \frac{2}{n} = H(X) + \frac{2}{n},$$

which approaches $H(X)$ as $n \to \infty$. No block coding is needed; the overhead is only 2 bits per message (not per symbol), making arithmetic coding essentially optimal for any source and any message length.

Precision and implementation. A naive implementation requires real-number arithmetic with precision $n \log(1/p_{\min})$ bits, which grows with n. The practical solution is *renormalization*: maintain ℓ and h as integer approximations in a fixed precision register (typically 32 or 64 bits), and "emit" the most significant bits as they become determined. When ℓ and h both start with the same bit b, emit b and shift both left (i.e. double the interval to $[2\ell, 2h)$ or $[2\ell - 1, 2h - 1)$ depending on b). The *E3 scaling* condition handles the rare case where $\ell \in [0.25, 0.5)$ and $h \in [0.5, 0.75)$ (the interval straddles 0.5 but neither half has been determined). After renormalization, the register maintains a sliding window into the arithmetic of the exact interval.

Example — Arithmetic coding of "BAC" over alphabet $\{A, B, C\}$

Source: $p(A) = 0.5, p(B) = 0.25, p(C) = 0.25$. Cumulative: $F_A = [0, 0.5)$, $F_B = [0.5, 0.75)$, $F_C = [0.75, 1.0)$.
Encode "BAC":

- $[\ell, h) = [0, 1)$.
- Symbol B: $\ell \leftarrow 0 + 1 \cdot 0.5 = 0.5$; $h \leftarrow 0 + 1 \cdot 0.75 = 0.75$. $[\ell, h) = [0.5, 0.75)$.
- Symbol A: $\ell \leftarrow 0.5 + 0.25 \cdot 0 = 0.5$; $h \leftarrow 0.5 + 0.25 \cdot 0.5 = 0.625$. $[\ell, h) = [0.5, 0.625)$.
- Symbol C: $\ell \leftarrow 0.5 + 0.125 \cdot 0.75 = 0.59375$; $h \leftarrow 0.5 + 0.125 \cdot 1.0 = 0.625$. $[\ell, h) = [0.59375, 0.625)$.

Interval length: $0.625 - 0.59375 = 0.03125 = 0.25 \times 0.5 \times 0.25 = p(B)p(A)p(C)$. ✓
Output: a number in $[0.59375, 0.625)$. Taking $0.625 = 0.101_2$ (binary): the codeword 101 lies in the interval (verify: $0.101_2 = 0.625 \in [0.59375, 0.625]$ — borderline; use $0.1001_2 = 0.5625$ which is in $[0.59375, 0.625)$). So output 1001 (4 bits).
Theoretical length: $-\log_2(p(B)p(A)p(C)) = -\log_2(0.03125) = 5$ bits, achieved to within $\lceil 5 \rceil + 1 = 6$ bits. Entropy: $H = 0.5 \times 1 + 0.25 \times 2 + 0.25 \times 2 = 1.5$ bits/symbol; message entropy $= 4.5$ bits; 4 bits is within 1 bit. For longer messages, the per-symbol overhead vanishes.

4.3 Universal Source Coding: The Lempel-Ziv Family

Huffman and arithmetic coding require knowledge of the source probability distribution p. *Universal codes* achieve compression approaching the entropy rate $H(\mathcal{X})$ of any stationary ergodic source, without knowledge of the source, by learning the source model from the data. The Lempel-Ziv family of algorithms, introduced in two landmark papers (1977 and 1978), are the most widely used universal compressors (underlying DEFLATE, gzip, LZ4, and Zstandard).

LZ77 Algorithm

Input: String $s_1 s_2 \cdots s_n$.
Maintain: A sliding window of the last W symbols (the *dictionary*).
Encode: At position t, find the longest match of $s_t s_{t+1} \cdots$ in the window $s_{t-W} \cdots s_{t-1}$. Output a triple $(\text{offset}, \text{length}, \text{next character})$, where offset is the position of the match in the window, length is the match length, and the next character is $s_{t+\text{length}}$ (the first non-matching symbol).
Advance: Move to position $t + \text{length} + 1$ and repeat.

LZ78 / LZW Algorithm

Maintain: A *dictionary* D of strings encountered so far, initialized with single characters of the alphabet.
Encode: Read input until the current string w is not in D. Output the code for the longest prefix of w that is in D, and add w to D as a new entry. Advance past $|w|$ symbols.
In LZW (a variant due to Welch), the output is simply the dictionary index of the longest matching prefix, with implicit addition of the new entry.

Universality theorem for LZ78. Let $X_1, X_2, \ldots$ be a stationary ergodic source with alphabet $\mathcal{X}$ and entropy rate $H = \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n)$. Let $c(X_1^n)$ denote the number of bits output by LZ78 to encode X_1^n. Then

$$\frac{c(X_1^n)}{n} \xrightarrow{n\to\infty} H \quad \text{almost surely.}$$

Proof sketch. The LZ78 parsing of X_1^n produces c_n phrases $w_1, w_2, \ldots, w_{c_n}$, each a new dictionary entry. By the recurrence structure, $c_n \approx n/\log n$ (each phrase uses roughly $\log n$ bits as the dictionary grows). The key estimate is that the number of phrases satisfies $c_n \le nH/\log(nH) + o(n/\log n)$, implying the bit rate converges to H. The formal proof uses the *parsing lemma* (each new phrase adds one dictionary entry and the dictionary grows like $c_n \le 2n/\log n$) together with the ergodic theorem to control the frequency of patterns. ■

Practical considerations. LZ77/LZW are not optimal for every source: they are designed for stationary sources with long-range correlations (like text and binary code). For i.i.d. or nearly i.i.d. sources, arithmetic coding with a trained model is typically better. Modern compressors (LZ4, Zstandard) combine LZ77 matching with entropy coding of the match lengths and literals, achieving both near-optimal compression and high decoding throughput.

4.4 Rate-Distortion Theory

Rate-distortion theory is the information-theoretic framework for *lossy* compression: how many bits per source symbol are required if we allow a controlled amount of distortion? It is the counterpart of channel capacity for the source side.

Distortion Measure

A *distortion measure* (or *distortion function*) is a non-negative function $d : \mathcal{X} \times \hat{\mathcal{X}} \to [0, \infty)$, where $\mathcal{X}$ is the source alphabet and $\hat{\mathcal{X}}$ is the reconstruction alphabet. The distortion of a code that maps source sequence $\mathbf{x} = (x_1, \dots, x_n)$ to reconstruction $\hat{\mathbf{x}} = (\hat{x}_1, \dots, \hat{x}_n)$ is

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{t=1}^{n} d(x_t, \hat{x}_t).$$

The most common distortion measure is the *squared error*: $d(x, \hat{x}) = (x - \hat{x})^2$, yielding *mean squared error* (*MSE*).

Rate-Distortion Function

The *rate-distortion function* $R(D)$ for source $X \sim p_X$ and distortion measure d is

$$R(D) = \min_{p(\hat{x}|x) : \mathbb{E}[d(X,\hat{X})] \le D} I\left(X; \hat{X}\right),$$

where the minimization is over all conditional distributions (channels) $p(\hat{x}|x)$ such that the expected distortion $\mathbb{E}[d(X, \hat{X})] = \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x, \hat{x}) \le D$.

Properties of $R(D)$.

- $R(D)$ is non-negative, non-increasing, and convex in D.
- $R(0) = H(X)$ (lossless compression; no distortion allowed).
- $R(D) = 0$ for $D \ge D_{\max} = \min_{\hat{x}} \mathbb{E}[d(X, \hat{x})]$ (trivial compression to a single reconstruction symbol).
- $-R'(D) = 1/\lambda^*(D)$ where $\lambda^*(D)$ is the optimal Lagrange multiplier; this is the *slope* of the rate-distortion function.

Shannon's Rate-Distortion Theorem

For a memoryless source $X \sim p_X$ and distortion measure d, the *operational rate-distortion function* (the infimum of rates R such that distortion $\le D$ is achievable with block codes of growing block length) equals $R(D)$.
More precisely: for any $R > R(D)$, there exist $(2^{nR}, n)$ codes with distortion $\le D + \epsilon$ for sufficiently large n. Conversely, any $(2^{nR}, n)$ code with $R < R(D)$ must incur distortion $> D - \epsilon$.

Proof of achievability (sketch). Fix $p^*(\hat{x}|x)$ achieving $R(D)$. Generate 2^{nR} reconstruction codewords $\hat{\mathbf{X}}(1), \dots, \hat{\mathbf{X}}(2^{nR})$, each drawn i.i.d. from $p_{\hat{X}}(\hat{x}) = \sum_x p_X(x)p^*(\hat{x}|x)$. Given source sequence $\mathbf{x}$, encode to the index m such that $(\mathbf{x}, \hat{\mathbf{X}}(m))$ are jointly typical with respect to $p_X p^*$. By the conditional AEP and joint typicality, with high probability a jointly typical codeword exists (since $2^{nR} \ge 2^{nI(X;\hat{X})} = 2^{nR(D)}$), and the distortion of any jointly typical pair satisfies $\frac{1}{n}\sum_t d(x_t, \hat{x}_t) \le D + \epsilon$ by the law of large numbers. ■

Gaussian rate-distortion function. For a Gaussian source $X \sim \mathcal{N}(0, \sigma^2)$ with squared-error distortion $d(x, \hat{x}) = (x - \hat{x})^2$:

$$R(D) = \frac{1}{2} \log_2 \frac{\sigma^2}{D}, \quad 0 \le D \le \sigma^2.$$

Equivalently, $D(R) = \sigma^2 2^{-2R}$: each extra bit of rate halves the MSE. The achievability uses the test channel $\hat{X} = X + Z$ with $Z \sim \mathcal{N}(0, D)$ independent of X (optimal forward channel); the mutual information is $I\big(X; \hat{X}\big) = \frac{1}{2}\log_2(1 + \sigma^2/D) - \frac{1}{2}\log_2(1 + 1)$... more carefully, $R(D) = \frac{1}{2}\log_2(\sigma^2/D)$ which is the mutual information $I\big(X; \hat{X}\big)$ when $\hat{X} = X + Z$ with $Z \sim \mathcal{N}(0, D - D_{\hat{X}})$... the clean derivation uses the optimization: for fixed D, the optimal reconstruction is $\hat{X} = X -$ MMSE estimate $= X \cdot \frac{D}{\sigma^2} +$ noise, giving $I\big(X; \hat{X}\big) = h(X) - h(X|\hat{X}) = \frac{1}{2}\log(2\pi e \sigma^2) - \frac{1}{2}\log(2\pi e D) = \frac{1}{2}\log(\sigma^2/D)$.

Example — Rate-distortion for a Gaussian speech source

Suppose a speech signal is modeled as i.i.d. Gaussian with variance $\sigma^2 = 1$ (after normalization). At a rate of $R = 4$ bits/sample, the achievable distortion is $D = \sigma^2 2^{-2R} = 2^{-8} \approx 0.0039$, i.e. the MSE is less than 0.4% of the source variance. At $R = 8$ bits/sample (CD quality for a single channel), $D = 2^{-16} \approx 1.5 \times 10^{-5}$: extremely high fidelity. The rate-distortion curve shows a 6 dB improvement in SNR ($10\log_{10}(1/D)$ decreases by 6 dB) for each extra bit of rate — the same 6 dB/bit rule that appears in PCM quantization.

4.5 The Blahut-Arimoto Algorithm for Rate-Distortion

The rate-distortion function $R(D)$ is defined as an optimization over conditional distributions. In general, computing it requires solving a convex program (since $I(X; \hat{X})$ is jointly convex in $(p_X, p_{\hat{X}|X})$ and linear in $p_{\hat{X}|X}$ for fixed p_X). The *Blahut-Arimoto algorithm for rate-distortion* is an alternating minimization procedure analogous to the Blahut-Arimoto algorithm for channel capacity.

Blahut-Arimoto Rate-Distortion Algorithm

Input: Source distribution $p(x)$, distortion matrix $d(x, \hat{x})$, slope parameter $s < 0$. (The parameter s controls which point on the $R(D)$ curve is computed: $s = -\lambda$ where λ is the Lagrange multiplier for the distortion constraint.)
Initialize: $q(\hat{x}) =$ uniform over $\hat{\mathcal{X}}$.
Iterate until convergence:

$$\phi(x) \leftarrow \sum_{\hat{x}} q(\hat{x}) \exp(s \cdot d(x, \hat{x})), \tag{1}$$

$$p(\hat{x}|x) \leftarrow \frac{q(\hat{x}) \exp(s \cdot d(x, \hat{x}))}{\phi(x)}, \tag{2}$$

$$q(\hat{x}) \leftarrow \sum_{x} p(x) p(\hat{x}|x). \tag{3}$$

Output: Rate $R = \sum_{x,\hat{x}} p(x)p(\hat{x}|x) \log_2 \frac{p(\hat{x}|x)}{q(\hat{x})}$ and distortion $D = \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x, \hat{x})$.

Convergence and optimality. The algorithm performs coordinate descent on the Lagrangian

$$L(p(\hat{x}|x), q(\hat{x})) = \sum_{x,\hat{x}} p(x)p(\hat{x}|x) \log \frac{p(\hat{x}|x)}{q(\hat{x})} + s \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x, \hat{x}),$$

alternating between updating $p(\hat{x}|x)$ (which yields update (2)) and $q(\hat{x})$ (which yields update (3)). Each update decreases L; since L is bounded below, the algorithm converges. The fixed point satisfies the KKT conditions for the rate-distortion optimization problem.

Sweeping the $R(D)$ curve. By varying s from 0 to $-\infty$, one traces the entire rate-distortion curve. At $s = 0$, the algorithm converges to zero rate and maximum distortion. As $s \to -\infty$, it approaches zero distortion and maximum rate (entropy). Each value of s corresponds to a tangent to the $R(D)$ curve with slope $1/s$ (or, in the (D, R) plane, slope s).

Part 5 — Kolmogorov Complexity

Computability prerequisites: Turing machines, universal Turing machines, undecidability. Kolmogorov complexity $K(x)$: definition, invariance theorem. Basic properties: subadditivity, $K(x) \leq |x| + O(1)$, most strings are incompressible. The universal semi-measure and Solomonoff induction. The incompressibility method: applications in combinatorics. Algorithmic mutual information and conditional complexity. Connections to Shannon entropy: the AEP revisited. Logical depth and resource-bounded complexity.

5.1 Computability Prerequisites

Kolmogorov complexity is a non-probabilistic, algorithmic notion of information content: the complexity of a string x is the length of the shortest computer program that outputs x. To make this precise, we need a brief review of computability theory.

Turing Machine and Universal Turing Machine

A *Turing machine* (TM) T is a finite automaton with an infinite tape, capable of reading, writing, and moving along the tape. A TM T *computes* a (partial) function $T : \{0,1\}^* \to \{0,1\}^*$ in the sense that on input p, T either halts with output $T(p) = x$, or runs forever.
A *universal Turing machine* (UTM) $\mathcal{U}$ is a TM that simulates any TM: $\mathcal{U}(\langle T, p\rangle) = T(p)$ for all TMs T and inputs p, where $\langle T, p\rangle$ denotes an encoding of T and p as a single binary string. Such $\mathcal{U}$ exist (this is Turing's universality theorem, 1936).

Kolmogorov Complexity

Fix a universal Turing machine $\mathcal{U}$. The *Kolmogorov complexity* of a binary string $x \in \{0,1\}^*$ is

$$K_{\mathcal{U}}(x) = \min\{|p| : \mathcal{U}(p) = x\},$$

the length of the shortest binary program p such that $\mathcal{U}$ outputs x. (If no such p exists, $K_{\mathcal{U}}(x) = \infty$, but this does not occur for any computable x.)

The invariance theorem. Different choices of UTM $\mathcal{U}$ and $\mathcal{U}'$ change K by at most a constant:

$$|K_{\mathcal{U}}(x) - K_{\mathcal{U}'}(x)| \leq c_{\mathcal{U},\mathcal{U}'},$$

where c depends only on $\mathcal{U}$ and $\mathcal{U}'$, not on x. This is because $\mathcal{U}$ can simulate $\mathcal{U}'$ given $\mathcal{U}'$'s description (a fixed overhead $O(1)$). The invariance theorem justifies writing $K(x)$ without specifying $\mathcal{U}$, since all choices agree to within a constant.

5.2 Basic Properties of Kolmogorov Complexity

Conditional Kolmogorov Complexity

The *conditional complexity* of x given y is

$$K(x|y) = \min\{|p| : \mathcal{U}(\langle p, y\rangle) = x\},$$

the length of the shortest program that outputs x when given y as a free input.

Basic inequalities.

- *Upper bound by description*: $K(x) \leq |x| + O(1)$, since the identity program ("output your input") is a valid description.
- *Subadditivity*: $K(x, y) \leq K(x) + K(y) + O(\log \min(K(x), K(y)))$. Roughly, we can describe the pair by describing each separately, with a constant overhead for separating the two descriptions (the log term accounts for self-delimiting encoding).
- *Chain rule*: $K(x, y) = K(x) + K(y|x) + O(\log K(x, y))$.
- *Symmetry of information*: $K(x) + K(y|x) = K(y) + K(x|y) + O(\log K(x, y))$. This is the algorithmic analogue of $H(X) + H(Y|X) = H(Y) + H(X|Y)$.

Most strings are incompressible. The number of binary strings of length n is 2^n. The number of programs of length $< n - c$ is $< 2^{n-c}$. So at most 2^{n-c} strings of length n can be compressed by more than c bits. In particular, at least $2^n - 2^{n-c} = 2^n(1 - 2^{-c})$ strings of length n satisfy $K(x) \geq n - c$. Taking $c = 2$: at least 75% of n-bit strings are *incompressible* in the sense $K(x) \geq n - 2$.

Incompressible String

A string x is *c-incompressible* if $K(x) \geq |x| - c$. A string with $K(x) \geq |x|$ (no compression possible) is called a *Kolmogorov random* string.

Kolmogorov randomness and randomness testing. A string x is Kolmogorov random iff it passes every computable statistical test: no computable property that holds for a negligible fraction of n-bit strings can hold for x. This makes Kolmogorov complexity a universal measure of randomness, subsuming all classical statistical tests (runs, correlations, birthday paradoxes, etc.). The incompressible strings are, in a precise sense, the "most random" binary strings.

5.3 The Incompressibility Method

The *incompressibility method* uses Kolmogorov complexity as a proof technique in combinatorics and information theory: to lower-bound the size of some combinatorial object, show that any incompressible string of appropriate length requires that object to be large.

Example: lower bound on sorting complexity. Any comparison-based sorting algorithm on n elements must make $\Omega(n \log n)$ comparisons in the worst case. Proof via incompressibility:

Consider an incompressible permutation σ of $\{1, \ldots, n\}$, i.e. one with $K(\sigma) \geq \log_2(n!) - O(1) = n \log_2 n - O(n)$ (since there are $n!$ permutations and most are incompressible). Given the sequence of comparison outcomes (a binary string of length equal to the number of comparisons T), together with the algorithm, one can reconstruct σ (since sorting reveals the sorted order, from which σ is recovered by the inverse permutation). Therefore $T + O(\log n) \geq K(\sigma) \geq n \log_2 n - O(n)$, implying $T = \Omega(n \log n)$. ■

Example: the string complexity of the Fibonacci sequence. The Fibonacci string F_n (defined by $F_0 = 0, F_1 = 1, F_n = F_{n-1}F_{n-2}$, the concatenation recurrence) of length $|F_n| = F_n$ (the Fibonacci number) satisfies $K(F_n) = O(\log n)$: given n, the string F_n can be computed by a short program (the recurrence), so its Kolmogorov complexity is logarithmic in its length, far below $|F_n|$. This quantifies the sense in which the Fibonacci string is highly structured and compressible.

5.4 The Universal Prior and Solomonoff Induction

One of the deepest applications of Kolmogorov complexity is to the problem of *universal induction*: given a sequence of observations, what is the optimal prediction of the next symbol without any prior assumption on the source?

Universal Semi-Measure

The *universal a priori semi-measure* M is defined as

$$M(x) = \sum_{T : T(\epsilon) \text{ starts with } x} 2^{-|T|},$$

where the sum is over all programs T (descriptions for the UTM $\mathcal{U}$) such that $\mathcal{U}(T)$ outputs a string starting with x. (This is a semi-measure because it may not sum to 1 over all x of a given length, due to non-halting programs.)

The coding theorem. The fundamental relation between Kolmogorov complexity and the universal semi-measure is:

$$M(x) = 2^{-K(x)+O(1)},$$

i.e. the negative log of the universal semi-measure equals Kolmogorov complexity, up to a constant. This is the *algorithmic coding theorem*, the Kolmogorov-complexity analogue of the Shannon coding theorem.

Solomonoff induction. Given a sequence $x_1, \dots, x_n$, predict x_{n+1} using the universal a priori probability:

$$P_{\text{Solomonoff}}(x_{n+1} = b \mid x_1, \dots, x_n) = \frac{M(x_1 \cdots x_n b)}{M(x_1 \cdots x_n)}.$$

This is the *Bayesian* prediction under the universal prior M. It is *universally optimal* in the following sense: for any computable measure μ on binary sequences, $\mu(x_1 \cdots x_n) \leq M(x_1 \cdots x_n) \cdot 2^{O(1)}$, so M dominates (up to a constant) every computable prior. Therefore Solomonoff induction makes the best possible predictions for any computable source, achieving the minimum cumulative prediction error.

The connection to minimum description length (MDL). The *minimum description length principle* (Rissanen, 1978) translates Solomonoff induction into a model selection framework: among all models $\mathcal{M}_1, \mathcal{M}_2, \dots$ for the data x^n, select the one that minimizes the *description length*

$$\text{MDL}(\mathcal{M}_i) = K(\mathcal{M}_i) + K(x^n | \mathcal{M}_i),$$

the sum of the complexity of the model and the complexity of the data given the model. This is an algorithmic implementation of Occam's Razor and a non-probabilistic precursor to Bayesian model selection. MDL is extensively used in statistics, machine learning, and information-theoretic approaches to hypothesis testing.

5.5 Connections to Shannon Entropy: The AEP Revisited

The AEP from the Kolmogorov perspective. For an i.i.d. source $X_1, \dots, X_n$ with distribution p and entropy $H(X)$, almost all strings $x^n = x_1 \cdots x_n$ are *typical* with probability close to 2^{-nH}. The Kolmogorov-complexity perspective gives a complementary characterization: most x^n are such that $K(x^n) \approx nH(X)$, i.e. the shortest description of a typical string has approximately $nH(X)$ bits.

More precisely, for $\epsilon > 0$ and sufficiently large n:

$$\Pr[|K(X_1^n) - nH(X)| > n\epsilon] \to 0 \quad \text{as } n \to \infty.$$

This is the *algorithmic AEP*, an extension of the classical AEP. The proof direction $K(X_1^n) \lesssim nH(X)$ follows because a near-optimal arithmetic code provides a description of length $\approx nH(X)$. The direction $K(X_1^n) \gtrsim nH(X)$ follows from the counting argument (most strings cannot be compressed below their entropy).

Shannon entropy as expected Kolmogorov complexity. For a finite-valued random variable $X \sim p$:

$$H(X) = \mathbb{E}[K(X)] + O(1).$$

More precisely, $H(X) - O(\log H(X)) \leq \mathbb{E}[K(X)] \leq H(X) + O(1)$. This shows that Shannon entropy and Kolmogorov complexity agree on average: the expected shortest description length of a randomly drawn symbol equals (to within constants) the Shannon entropy. The Kolmogorov complexity is thus a *single-instance* version of Shannon entropy — it applies to individual strings, not distributions, while Shannon entropy is an average quantity.

Logical depth and computational complexity. Not all complex objects are *random*: a string may have high Kolmogorov complexity but be easily computable from a short program that runs in exponential time. Charles Bennett's *logical depth* measures the *running time* of the shortest program:

$$\text{depth}_s(x) = \min\{t : \exists p, \mathcal{U}(p) = x \text{ in time } t, |p| \leq K(x) + s\},$$

the minimum computation time among near-shortest programs. Logically deep strings require a long computation to produce even from their optimal description, capturing the intuition that a thermodynamic state with high entropy was produced by a long evolutionary process (e.g. a protein, a crystal, a developed organism) rather than being trivially random.

Numerical Tour: Part 1 — Sphere-Packing and Weight Enumerators

In this tour we work through concrete calculations with the Hamming bound, the GV bound, and weight enumerators for small codes.

Hamming bound at various parameters. For a binary $(n, k, 3)$ code: since $V(n, 1) = 1 + n$, the Hamming bound gives $k \leq n - \log_2(1 + n)$.

n	$\log_2(1+n)$	Hamming bound on k	Achieved by
3	2.00	1	Hamming $[3, 1, 3]$ (repetition)
7	3.00	4	Hamming $[7, 4, 3]$
15	4.00	11	Hamming $[15, 11, 3]$
31	5.00	26	Hamming $[31, 26, 3]$
63	6.00	57	Hamming $[63, 57, 3]$

GV bound for $d = 5$. The GV bound asserts there exists an $(n, k, 5)$ code with $k \geq n - \log_2 V(n, 4)$. $V(n, 4) = 1 + n + \binom{n}{2} + \binom{n}{3} + \binom{n}{4}$.

For $n = 21$: $V(21, 4) = 1 + 21 + 210 + 1330 + 5985 = 7547 < 2^{13}$, so $k \geq 21 - 13 = 8$. A $(21, 8, 5)$ code exists (the GV bound guarantees it). The Hamming bound gives $k \leq 21 - \log_2(V(21, 2)) = 21 - \log_2(232) \approx 21 - 7.86 = 13.14$, so $k \leq 13$. The GV bound says $k \geq 8$; the Hamming bound says $k \leq 13$. The true maximum, $A(21, 5)$, is between these.

Weight enumerator of the $[7, 4, 3]$ Hamming code. We established $W(x, y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7$. Its dual code is $[7, 3, 4]$, the simplex code (also called the first-order Reed-Muller code). By the MacWilliams identity:

$$W_{\mathcal{C}^\perp}(x, y) = \frac{1}{16} W_{\mathcal{C}}(x + y, x - y).$$

Compute $W_{\mathcal{C}}(x + y, x - y)$:

$$(x + y)^7 + 7(x + y)^4(x - y)^3 + 7(x + y)^3(x - y)^4 + (x - y)^7.$$

After expanding (using binomial theorem) and dividing by 16, one obtains: $W_{\mathcal{C}^\perp}(x, y) = x^7 + 7x^3y^4$. This says the dual $[7, 3, 4]$ code (of size 8) has 1 codeword of weight 0 and 7 codewords of weight 4. Verify: the $[7, 3, 4]$ simplex code indeed has minimum weight 4 (the weight of its minimum-weight codeword) and every nonzero codeword has weight 4. ✓

Numerical Tour: Part 2 — Reed-Solomon Encoding and Decoding

RS$(5, 3)$ over $\mathbb{F}_7$. Take $\mathbb{F}_7 = \{0, 1, 2, 3, 4, 5, 6\}$ with arithmetic mod 7. Choose evaluation points $\alpha_1, \ldots, \alpha_5 = 1, 2, 3, 4, 5$. The RS$(5, 3)$ code maps degree-≤ 2 polynomials to their evaluations:

Information word $\mathbf{u} = (1, 3, 2)$ corresponds to $f(x) = 1 + 3x + 2x^2$.

Evaluations (mod 7):

$$\begin{aligned}
f(1) &= 1 + 3 + 2 = 6 \\
f(2) &= 1 + 6 + 8 = 1 + 6 + 1 = 8 \equiv 1 \pmod 7 \\
f(3) &= 1 + 9 + 18 = 1 + 2 + 4 = 7 \equiv 0 \pmod 7 \\
f(4) &= 1 + 12 + 32 = 1 + 5 + 4 = 10 \equiv 3 \pmod 7 \\
f(5) &= 1 + 15 + 50 = 1 + 1 + 1 = 3 \pmod 7
\end{aligned}$$

Codeword: $(6,1,0,3,3)$. Parameters: $[5,3,3]$ over $\mathbb{F}_7$.

Decoding with one error. Suppose the received word is $\mathbf{y} = (6,1,0,3,5)$ (error in position 5: $3 \to 5$, i.e. $e_5 = 2$). Syndromes: $S_j = \sum_{i=1}^{5} y_i \alpha_i^j$.

$S_0 = 6+1+0+3+5 = 15 \equiv 1 \pmod 7$. $S_1 = 6\cdot 1+1\cdot 2+0\cdot 3+3\cdot 4+5\cdot 5 = 6+2+0+12+25 = 45 \equiv 3 \pmod 7$. $S_2 = 6\cdot 1+1\cdot 4+0\cdot 9+3\cdot 16+5\cdot 25 = 6+4+0+48+125 = 183 \equiv 183-26\cdot 7 = 183-182 = 1 \pmod 7$.

For one error at position $i_1 = 5$ with magnitude $e_5 = 2$, the Berlekamp-Massey algorithm finds $\Lambda(z) = 1 - 5^{-1}e_5 z = 1 - 3\cdot 2\cdot z = 1 - 6z = 1 + z \pmod 7$; note $5^{-1} = 3$ since $3\cdot 5 = 15 \equiv 1$)... the computation proceeds via Newton's identity $S_1 + \Lambda_1 S_0 = 0 \Rightarrow \Lambda_1 = -S_1/S_0 = -3\cdot 1^{-1} = -3 \equiv 4 \pmod 7$. Error locator: $\Lambda(z) = 1 + 4z$. Root: $\Lambda(\hat{z}) = 0 \Rightarrow \hat{z} = -4^{-1} \equiv -2^{-1} \equiv -(4) \equiv 3 \pmod 7$. Since $\alpha_5^{-1} = 5^{-1} = 3$, the error is in position 5. Forney formula: $e_5 = S_0/\Lambda'(\alpha_5^{-1}) \cdots = 2$ (the original error magnitude). Corrected word: $(6,1,0,3,3) = \mathbf{c}$. ✓

Numerical Tour: Part 3 — LDPC Belief Propagation on the BEC

A $(3,6)$-regular LDPC code, small example. Consider a small Tanner graph with $n = 6$ variable nodes, $m = 3$ check nodes, parity-check matrix:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Each row has degree 3, each column has degree... variable nodes 1 and 3 have degree 2; variable nodes 2 has degree 2; etc. (This is not regular, but illustrative.)

Suppose transmitted codeword $\mathbf{c} = (1,0,1,0,1,0)$ (verify: $\mathbf{Hc}^\top = (1+0+0, 0+0+0, 1+0+0)^\top = (1,0,1)^\top \neq \mathbf{0}$; so we should find the actual codewords. The null space of $\mathbf{H}$ over $\mathbb{F}_2$ has dimension 3. Let us just trace the algorithm conceptually.)

Over the BEC with erasure probability $\epsilon = 0.4$: suppose bits $2,4,6$ are erased. Channel output: $(1,?,1,?,1,?)$.

Iteration 1. Variable-to-check messages: known bits send their values; erased bits send ?. $\mu_{v_1\to c_1} = 1$, $\mu_{v_2\to c_1} = ?$, $\mu_{v_4\to c_1} = ?$ (check c_1 involves v_1, v_2, v_4). $\mu_{v_2\to c_2} = ?$, $\mu_{v_3\to c_2} = 1$, $\mu_{v_5\to c_2} = 1$. $\mu_{v_1\to c_3} = 1$, $\mu_{v_3\to c_3} = 1$, $\mu_{v_6\to c_3} = ?$.

Check-to-variable messages: c_2: incoming from $v_3 = 1$, $v_5 = 1$; send to v_2: $1 \oplus 1 = 0 \to \mu_{c_2\to v_2} = 0$. c_3: incoming from $v_1 = 1$, $v_3 = 1$; send to v_6: $1 \oplus 1 = 0 \to \mu_{c_3\to v_6} = 0$. c_1: incoming from $v_1 = 1$, $v_4 = ?$, $v_2 = ?$; send to v_2: unknown (two unknowns). $\mu_{c_1\to v_4} = ?$.

Iteration 2. v_2 has received $\mu_{c_2\to v_2} = 0$: bit 2 is recovered as 0. v_6 has received $\mu_{c_3\to v_6} = 0$: bit 6 is recovered as 0. Now $v_2 = 0$ is known. Check c_1: incoming from $v_1 = 1$, $v_2 = 0$; send to v_4: $\mu_{c_1\to v_4} = 1 \oplus 0 = 1$. Bit $v_4 = 1$ recovered.

All erasures resolved in 2 iterations. ✓

Numerical Tour: Part 4 — Arithmetic Coding and Rate-Distortion

Arithmetic coding with a 3-symbol source. Source: $p(A) = 0.7, p(B) = 0.2, p(C) = 0.1$. Entropy: $H = 0.7\log_2(1/0.7) + 0.2\log_2(1/0.2) + 0.1\log_2(1/0.1) = 0.360 + 0.464 + 0.332 = 1.156$ bits/symbol.

Encode message "AAB" (length 3, theoretical length $\approx 3 \times 1.156 = 3.47$ bits).

$[\ell, h) = [0, 1)$.

- Symbol A ($[0, 0.7)$): $[\ell, h) = [0, 0.7)$.
- Symbol A ($[0, 0.7)$): $\ell = 0 + 0.7 \cdot 0 = 0$; $h = 0 + 0.7 \cdot 0.7 = 0.49$. $[0, 0.49)$.
- Symbol B ($[0.7, 0.9)$): $\ell = 0 + 0.49 \cdot 0.7 = 0.343$; $h = 0 + 0.49 \cdot 0.9 = 0.441$. $[0.343, 0.441)$.

Interval length $= 0.441 - 0.343 = 0.098 = p(A)^2 p(B) = 0.49 \times 0.2 = 0.098$. ✓

Binary code: need to represent a number in $[0.343, 0.441)$. $0.375 = 0.011_2 \in [0.343, 0.441)$. Codeword: 011 (3 bits). Ideal: $-\log_2(0.098) \approx 3.35$ bits; we used 3 bits. Excellent.

Gaussian rate-distortion numerical example. $X \sim \mathcal{N}(0, 1)$ ($\sigma^2 = 1$). Target: MSE $D = 0.1$. Required rate: $R(0.1) = \frac{1}{2}\log_2(1/0.1) = \frac{1}{2}\log_2(10) \approx 1.66$ bits/sample.

At $R = 2$ bits/sample, achievable distortion: $D = 2^{-2R} = 2^{-4} = 0.0625$. This means: with 2 bits per Gaussian sample (e.g., from a speech or audio signal with unit variance), the mean squared reconstruction error is 6.25% of the variance, or equivalently an SNR of $10\log_{10}(1/0.0625) = 12$ dB.

Formula Reference: Part 1 — Block Codes

Key Bounds Summary

For binary (n, k, d) codes with $t = \lfloor (d-1)/2 \rfloor$, $\delta = d/n$:

$$\text{Hamming: } k \le n - \log_2 V(n, t), \quad V(n, r) = \sum_{j=0}^{r} \binom{n}{j}.$$

$$\text{Singleton: } k \le n - d + 1.$$

$$\text{Plotkin: } A(n, d) \le 2d \text{ when } d > n/2.$$

$$\text{GV: } k \ge n - \log_2 V(n, d-1).$$

$$\text{Asymptotic Hamming: } R \le 1 - h_b(\delta/2).$$

$$\text{Asymptotic GV: } R \ge 1 - h_b(\delta), \quad 0 \le \delta \le 1/2.$$

Formula Reference: Part 2 — Linear and Cyclic Codes

Linear Code Summary

An $[n, k, d]_q$ linear code has:

- Generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$: $\mathcal{C} = \{\mathbf{uG} : \mathbf{u} \in \mathbb{F}_q^k\}$.
- Parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$: $\mathcal{C} = \ker_{\mathbb{F}_q}(\mathbf{H})$.
- $\mathbf{GH}^\top = \mathbf{0}$, $d_{\min}$ = min-weight column span of $\mathbf{H}$.
- Systematic: $\mathbf{G} = [\mathbf{I}_k | \mathbf{P}]$, $\mathbf{H} = [\mathbf{P}^\top | \mathbf{I}_{n-k}]$.
- Dual code $\mathcal{C}^\perp$: $[n, n-k]$ code with generator matrix $\mathbf{H}$.
- MacWilliams: $W_{\mathcal{C}^\perp}(x, y) = \frac{1}{|\mathcal{C}|} W_{\mathcal{C}}(x + y, x - y)$.
- RS(n, k) over $\mathbb{F}_q$: $[n, k, n-k+1]$, MDS, Vandermonde generator matrix.

- BCH bound: $g(x)$ with $\delta-1$ consecutive roots $\Rightarrow d \geq \delta$.

Formula Reference: Part 3 — LDPC and Turbo Codes

Iterative Codes Summary

LDPC:

- Tanner graph: variable nodes V, check nodes C, edge iff $H_{ij}=1$.
- (d_v, d_c)-regular: rate $R = 1 - d_v/d_c$.
- Degree distributions: $\lambda(x) = \sum_i \lambda_i x^{i-1}$, $\rho(x) = \sum_i \rho_i x^{i-1}$.
- BP update (LLR): $L_{v\to c}^{(\ell)} = L^{\text{ch}} + \sum_{c'\neq c} L_{c'\to v}^{(\ell-1)}$.
- Check update: $L_{c\to v}^{(\ell)} = 2\operatorname{arctanh}\Big(\prod_{v'\neq v} \tanh(L_{v'\to c}^{(\ell)}/2)\Big)$.
- BEC density evolution: $p_{\ell+1} = \epsilon\lambda(1-\rho(1-p_\ell))$.

Turbo codes:

- BCJR: $\alpha_t(s) = \sum_{s'} \alpha_{t-1}(s')\gamma_t(s',s)$; β backwards; APP $\propto \alpha\gamma\beta$.
- Extrinsic LLR: $L_e = L_{\text{post}} - L^{\text{ch}} - L_a$.
- Iterate: Decoder 1 $\to$ interleave extrinsic $\to$ Decoder 2 $\to$ de-interleave $\to$ repeat.

Formula Reference: Part 4 — Data Compression

Compression Summary

Huffman: $H(X) \leq \bar{L}_{\text{Huffman}} < H(X)+1$.
Arithmetic coding: Per-symbol overhead $\to 0$ as message length $n\to\infty$.
LZ universality: For ergodic source with rate H, $c(X_1^n)/n \to H$ a.s.
Rate-distortion:

$$R(D) = \min_{p(\hat{x}|x):\,\mathbb{E}d\leq D} I(X;\hat{X}), \quad D(R) = \max_{p(\hat{x}|x):\,I(X;\hat{X})\leq R} \mathbb{E}[d(X,\hat{X})].$$

Gaussian (σ^2, squared error): $R(D) = \frac{1}{2}\log_2(\sigma^2/D)$, $D(R) = \sigma^2 2^{-2R}$.
Blahut-Arimoto (R-D): alternate $\phi(x) = \sum_{\hat{x}} q(\hat{x})e^{sd(x,\hat{x})}$, $p(\hat{x}|x) = q(\hat{x})e^{sd}/\phi(x)$, $q(\hat{x}) = \sum_x p(x)p(\hat{x}|x)$.

Formula Reference: Part 5 — Kolmogorov Complexity

Kolmogorov Complexity Summary

- $K(x) = \min\{|p| : \mathcal{U}(p) = x\}$; invariant up to $O(1)$ across UTMs.
- Upper bound: $K(x) \leq |x| + O(1)$.
- Chain rule: $K(x,y) = K(x) + K(y|x) + O(\log K(x,y))$.
- Symmetry: $K(x) + K(y|x) = K(y) + K(x|y) + O(\log)$.
- Incompressibility: $|\{x\in\{0,1\}^n : K(x) < n-c\}| < 2^{n-c}$.
- Coding theorem: $M(x) = 2^{-K(x)+O(1)}$ (universal semi-measure).
- Algorithmic AEP: $K(X_1^n)/n \to H(X)$ in probability.

- $H(X) = \mathbb{E}[K(X)] + O(1)$ (Shannon = expected Kolmogorov).

Computational Notes: Algorithms and Implementation

Algorithm 1: Huffman Code Construction

Input: Probability vector $(p_1, \dots, p_M)$ with $\sum_i p_i = 1$.

Data structure: Priority queue Q (min-heap) keyed by probability.

Algorithm:

1. Initialize: for each i, push leaf node $\{p_i, \text{label} = x_i\}$ onto Q.
2. While $|Q| > 1$:
   (a) $n_1 \leftarrow Q$.pop_min(); $n_2 \leftarrow Q$.pop_min().
   (b) Create new internal node n with $n.p = n_1.p + n_2.p$; $n.children = (n_1, n_2)$.
   (c) Push n onto Q.
3. The remaining node is the root of the Huffman tree.
4. Assign codewords by depth-first traversal: append 0 going left, 1 going right.

Complexity: $O(M \log M)$ using a heap (each of the $M-1$ merge operations costs $O(\log M)$).

Canonical Huffman codes. A *canonical* Huffman code assigns codewords in a canonical order: for each length ℓ, symbols are assigned the numerically smallest binary strings of length ℓ not yet used, with shorter codewords always being numerically smaller than longer ones. Canonical codes can be represented by just the length sequence $(l_1, \dots, l_M)$, enabling faster decoding via a lookup table.

Algorithm 2: Arithmetic Encoder with Integer Arithmetic

Input: Symbol sequence $a_1, \dots, a_n$; cumulative probability table $F_0 = 0, F_1, \dots, F_M = 1$ as integers with denominator D (so $F_i \in \{0, \dots, D\}$).

Parameters: Precision $P = 2^B$ (typically $B = 32$); Half $= P/2$; Quarter $= P/4$.

State: $\ell = 0$; $h = P - 1$; underflow counter $f = 0$; output bit stream B.

For each symbol a_t:

1. $w \leftarrow h - \ell + 1$.
2. $h \leftarrow \ell + \lfloor w \cdot F_{a_t} / D \rfloor - 1$.
3. $\ell \leftarrow \ell + \lfloor w \cdot F_{a_t - 1} / D \rfloor$.
4. **Renormalization loop**: while $h <$ Half or $\ell \geq$ Half:
   (a) If $h <$ Half: emit bit 0 followed by f ones; $f \leftarrow 0$; $\ell \leftarrow 2\ell$; $h \leftarrow 2h + 1$.
   (b) If $\ell \geq$ Half: emit bit 1 followed by f zeros; $f \leftarrow 0$; $\ell \leftarrow 2(\ell - \text{Half})$; $h \leftarrow 2(h - \text{Half}) + 1$.
   (c) (E3 condition): if $\ell \geq$ Quarter and $h < 3 \cdot$Quarter: $f \leftarrow f + 1$; $\ell \leftarrow 2(\ell - \text{Quarter})$; $h \leftarrow 2(h - \text{Quarter}) + 1$.

Flush: emit final bits to determine $[\ell, h)$ uniquely.

Algorithm 3: Berlekamp-Massey for RS Decoding

Input: Syndrome sequence $S = (S_0, S_1, \dots, S_{2t-1})$ over $\mathbb{F}_q$.

Output: Error-locator polynomial $\Lambda(z) = 1 + \Lambda_1 z + \cdots + \Lambda_L z^L$.

1. Initialize: $\Lambda \leftarrow [1]$; $B \leftarrow [1]$; $L \leftarrow 0$; $b \leftarrow 1$; $m \leftarrow 1$.
2. For $n = 0, 1, \dots, 2t-1$:
   (a) $\delta \leftarrow S_n + \sum_{j=1}^{L} \Lambda_j S_{n-j} \pmod q$.
   (b) If $\delta = 0$: $m \leftarrow m + 1$.
   (c) If $\delta \neq 0$ and $2L \leq n$: $T \leftarrow \Lambda$; $\Lambda(z) \leftarrow \Lambda(z) - (\delta b^{-1}) z^m B(z)$; $B \leftarrow T$; $L \leftarrow n+1-L$; $b \leftarrow \delta$; $m \leftarrow 1$.
   (d) If $\delta \neq 0$ and $2L > n$: $\Lambda(z) \leftarrow \Lambda(z) - (\delta b^{-1}) z^m B(z)$; $m \leftarrow m + 1$.
3. Return Λ. Degree of Λ is L, the number of errors.

Chien search (finding roots): Evaluate $\Lambda(\alpha^{-j})$ for $j = 0, 1, \dots, n-1$. If $\Lambda(\alpha^{-j}) = 0$, position j is an error location.

Forney formula (error magnitudes): For each error position j with $X_j = \alpha^j$:

$$e_j = -\frac{X_j \Omega(X_j^{-1})}{\Lambda'(X_j^{-1})},$$

where $\Omega(z) = S(z)\Lambda(z) \pmod{z^{2t}}$ is the *error evaluator polynomial* and Λ' is the formal derivative of Λ.

Algorithm 4: Sum-Product (Belief Propagation) for LDPC

Input: Channel LLRs $L^{\text{ch}} = (L_1^{\text{ch}}, \dots, L_n^{\text{ch}})$; Tanner graph G; max iterations T.

Initialize: $L_{v_j \to c_i}^{(0)} \leftarrow L_j^{\text{ch}}$ for all edges (v_j, c_i).

For $\ell = 1, 2, \dots, T$:

1. **Check-to-variable update**: for each check node c_i and each $v_j \in \mathcal{N}(c_i)$:

$$L_{c_i \to v_j}^{(\ell)} \leftarrow 2 \operatorname{arctanh}\left(\prod_{v_{j'} \in \mathcal{N}(c_i) \setminus v_j} \tanh\left(\frac{L_{v_{j'} \to c_i}^{(\ell-1)}}{2} \right) \right).$$

2. **Variable-to-check update**: for each variable node v_j and each $c_i \in \mathcal{N}(v_j)$:

$$L_{v_j \to c_i}^{(\ell)} \leftarrow L_j^{\text{ch}} + \sum_{c_{i'} \in \mathcal{N}(v_j) \setminus c_i} L_{c_{i'} \to v_j}^{(\ell)}.$$

3. **Bit APP**: $L_j^{(\ell)} \leftarrow L_j^{\text{ch}} + \sum_{c_i \in \mathcal{N}(v_j)} L_{c_i \to v_j}^{(\ell)}$.
4. **Hard decision**: $\hat{b}_j \leftarrow \mathbf{1}[L_j^{(\ell)} < 0]$.
5. **Early termination**: if $\mathbf{H}\hat{\mathbf{b}}^\top = \mathbf{0}$, stop.

Algorithm 5: Blahut-Arimoto for Rate-Distortion

Input: Source distribution $p(x)$; distortion matrix $d(x, \hat{x})$; slope $s < 0$; tolerance ϵ.

Initialize: $q(\hat{x}) \leftarrow 1/|\hat{\mathcal{X}}|$ for all $\hat{x}$.

Repeat until convergence (change in q less than ϵ):

1. $\phi(x) \leftarrow \sum_{\hat{x}} q(\hat{x}) \exp(s \cdot d(x, \hat{x}))$ for all x.
2. $p(\hat{x}|x) \leftarrow q(\hat{x}) \exp(s \cdot d(x, \hat{x}))/\phi(x)$ for all $(x, \hat{x})$.
3. $q(\hat{x}) \leftarrow \sum_{x} p(x)p(\hat{x}|x)$ for all $\hat{x}$.

Output:

$$D \leftarrow \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x, \hat{x}), \quad R \leftarrow \sum_{x,\hat{x}} p(x)p(\hat{x}|x) \log_2 \frac{p(\hat{x}|x)}{q(\hat{x})}.$$

Repeat for $s \in \{-\lambda : \lambda > 0\}$ to trace the $R(D)$ curve.

Convergence rate. The algorithm converges at a linear rate: the deficiency $R^{(t)} - R(D)$ decreases geometrically with a contraction factor $\rho < 1$ determined by the Hessian of the mutual information at the fixed point. The practical stopping criterion is $|R^{(t+1)} - R^{(t)}| < \epsilon$.

Glossary of Key Terms and Symbols

$[n, k, d]_q$	Linear code of block length n, dimension k, minimum distance d, over $\mathbb{F}_q$.
$A(n, d)$	Maximum number of binary codewords in a code of block length n and minimum distance $\geq d$.
AEP	Asymptotic Equipartition Property (Volume I; see also algorithmic AEP, Section 5.5).
APP	*A posteriori probability*: posterior distribution of a symbol given received data.
AWGN	Additive White Gaussian Noise channel.
BA algorithm	Blahut-Arimoto algorithm (for channel capacity or rate-distortion function).
BCH code	Bose-Chaudhuri-Hocquenghem code; binary cyclic code with designed distance δ.
BCJR algorithm	Bahl-Cocke-Jelinek-Raviv algorithm for APP decoding on a trellis.
BEC	Binary Erasure Channel with erasure probability ϵ.
belief propagation	See sum-product algorithm.
BSC	Binary Symmetric Channel with crossover probability ϵ.
$d_H(\cdot, \cdot)$	Hamming distance.
$d_{\min}$	Minimum distance of a code.
density evolution	Tracking the distribution of BP messages over iterations for random LDPC ensembles.
extrinsic information	Information gained by one decoder beyond what the other decoder already knew.
$\mathbb{F}_q$	Finite field (Galois field) with $q = p^m$ elements.
GV bound	Gilbert-Varshamov lower bound on the size of codes.
Hamming bound	Sphere-packing upper bound: $2^k V(n, t) \leq 2^n$.
Huffman code	Optimal prefix-free code minimizing expected code length.
$h_b(p)$	Binary entropy function: $h_b(p) = -p \log_2 p - (1-p) \log_2 (1-p)$.
$I(X; \hat{X})$	Mutual information between X and reconstruction $\hat{X}$.
incompressibility method	Proof technique using Kolmogorov complexity to derive combinatorial lower bounds.
interleaver	Permutation π in turbo code that breaks correlation between component decoders.
$K(x)$	Kolmogorov complexity of string x.
$K(x\|y)$	Conditional Kolmogorov complexity.

Kolmogorov random String x with $K(x) \geq |x|$; no compression possible.

Kraft inequality Necessary and sufficient condition for prefix-free codes: $\sum 2^{-\ell_i} \leq 1$.

LDPC Low-Density Parity-Check code (Gallager, 1963; MacKay-Neal, 1996).

LLR Log-likelihood ratio: $\log \frac{P(\text{bit}=0)}{P(\text{bit}=1)}$.

LZ77 / LZ78 Lempel-Ziv universal compression algorithms (1977, 1978).

$M(x)$ Universal semi-measure.

G Generator matrix of a linear code ($k \times n$).

H Parity-check matrix of a linear code ($(n-k) \times n$).

MacWilliams identity Relation between weight enumerators of dual codes.

MDS code Maximum-distance separable code: achieves Singleton bound $d = n - k + 1$.

MDL Minimum Description Length principle.

min-entropy $H_\infty(X) = -\log \max_x p(x)$; see Volume I Section 1.6.

min-sum algorithm Approximate belief propagation replacing tanh by min; efficient hardware implementation.

MSE Mean Squared Error: $\mathbb{E}[(X - \hat{X})^2]$.

prefix-free code Code in which no codeword is a prefix of another.

R Code rate (bits per channel use).

$R(D)$ Rate-distortion function.

Reed-Solomon code MDS code over $\mathbb{F}_q$ defined by polynomial evaluation: RS(n, k).

Singleton bound $k \leq n - d + 1$ for any q-ary (n, k, d) code.

Solomonoff induction Bayesian prediction under the universal semi-measure M.

sum-product algorithm Belief propagation on a factor graph; exact on trees.

syndrome $\mathbf{s} = \mathbf{H}\mathbf{y}$; zero for valid codewords, encodes error location.

Tanner graph Bipartite graph representation of LDPC code's parity-check matrix.

turbo code Parallel concatenated convolutional code with iterative BCJR decoding.

UTM Universal Turing Machine.

$V(n, r)$ Volume of a Hamming ball: $\sum_{j=0}^{r} \binom{n}{j}$.

$w_H(\cdot)$ Hamming weight.

$W_{\mathcal{C}}(x, y)$ Weight enumerator polynomial of code $\mathcal{C}$.

A Proof of the MacWilliams Identity

We give a complete proof of the MacWilliams identity. Let $\mathcal{C}$ be a binary linear $[n, k]$ code with dual $\mathcal{C}^\perp$. We use the Fourier analysis of the group $\mathbb{F}_2^n$.

Characters. The characters of $(\mathbb{F}_2^n, +)$ are the functions $\chi_{\mathbf{u}}(\mathbf{v}) = (-1)^{\mathbf{u}\cdot\mathbf{v}}$ for $\mathbf{u} \in \mathbb{F}_2^n$. Character orthogonality gives

$$\frac{1}{2^n} \sum_{\mathbf{v}\in\mathbb{F}_2^n} (-1)^{\mathbf{u}\cdot\mathbf{v}} = \mathbf{1}[\mathbf{u} = \mathbf{0}].$$

The annihilator of $\mathcal{C}$ under characters is exactly $\mathcal{C}^\perp$:

$$\sum_{\mathbf{c}\in\mathcal{C}} (-1)^{\mathbf{u}\cdot\mathbf{c}} = \begin{cases} |\mathcal{C}| & \mathbf{u} \in \mathcal{C}^\perp \\ 0 & \text{otherwise.} \end{cases}$$

The Hamming weight enumerator as a polynomial. Write $W_{\mathcal{C}}(x, y) = \sum_{\mathbf{c}\in\mathcal{C}} x^{n-w(\mathbf{c})} y^{w(\mathbf{c})} = \sum_{\mathbf{c}\in\mathcal{C}} \prod_{i=1}^n (x^{1-c_i} y^{c_i})$.

Now compute $W_{\mathcal{C}}(x + y, x - y)$:

$$W_{\mathcal{C}}(x + y, x - y) = \sum_{\mathbf{c}\in\mathcal{C}} \prod_{i=1}^n (x + y)^{1-c_i} (x - y)^{c_i}.$$

Use the substitution $(x+y)^{1-c_i}(x-y)^{c_i} = (x+y)\left(\frac{x-y}{x+y}\right)^{c_i}$; more usefully, note $(x+y)^{1-c_i}(x-y)^{c_i} = x + y \cdot (-1)^{c_i} \cdot$ (something)... cleaner: for each i,

$$(x + y)^{1-c_i}(x - y)^{c_i} = \sum_{b_i\in\{0,1\}} \binom{?}{} \ldots$$

The cleanest route: expand using binomial,

$$\prod_{i=1}^n (x + y)^{1-c_i}(x - y)^{c_i} = \prod_i \sum_{b_i\in\{0,1\}} \frac{(x + y)^{1-c_i}(x - y)^{c_i}}{1} \cdot \ldots$$

Let us use a different expansion. For any bit $c \in \{0, 1\}$: $(x + y)^{1-c}(x - y)^c = x + (-1)^c y$ (check: $c = 0$: $x + y$; $c = 1$: $x - y$. ✓).

Wait: $(x+y)^{1-c}(x-y)^c$. For $c = 0$: $(x+y)^1(x-y)^0 = x+y$. For $c = 1$: $(x+y)^0(x-y)^1 = x-y$. And $x + (-1)^c y$: $c = 0$: $x + y$; $c = 1$: $x - y$. ✓

Therefore

$$W_{\mathcal{C}}(x + y, x - y) = \sum_{\mathbf{c}\in\mathcal{C}} \prod_{i=1}^n (x + (-1)^{c_i} y).$$

Now expand this product over $\mathbf{b} \in \{0, 1\}^n$ (by choosing, for each factor, either x or $(-1)^{c_i} y$):

$$\prod_{i=1}^n (x + (-1)^{c_i} y) = \sum_{\mathbf{b}\in\{0,1\}^n} x^{n-|\mathbf{b}|} (-1)^{\mathbf{b}\cdot\mathbf{c}} y^{|\mathbf{b}|}.$$

(Factor i contributes x if $b_i = 0$, or $(-1)^{c_i} y$ if $b_i = 1$; the product over i gives $x^{n-|\mathbf{b}|} \cdot \prod_{i:b_i=1} (-1)^{c_i} \cdot y^{|\mathbf{b}|} = x^{n-|\mathbf{b}|}(-1)^{\mathbf{b}\cdot\mathbf{c}} y^{|\mathbf{b}|}$.)

Summing over $\mathbf{c} \in \mathcal{C}$:

$$W_{\mathcal{C}}(x+y, x-y) = \sum_{\mathbf{b}\in\{0,1\}^n} x^{n-|\mathbf{b}|} y^{|\mathbf{b}|} \underbrace{\sum_{\mathbf{c}\in\mathcal{C}} (-1)^{\mathbf{b}\cdot\mathbf{c}}}_{=|\mathcal{C}|\cdot\mathbf{1}[\mathbf{b}\in\mathcal{C}^\perp]} = |\mathcal{C}| \sum_{\mathbf{b}\in\mathcal{C}^\perp} x^{n-|\mathbf{b}|} y^{|\mathbf{b}|} = |\mathcal{C}| \cdot W_{\mathcal{C}^\perp}(x,y).$$

Dividing by $|\mathcal{C}| = 2^k$: $W_{\mathcal{C}^\perp}(x,y) = \frac{1}{|\mathcal{C}|} W_{\mathcal{C}}(x+y, x-y)$. ■

B Proof of Shannon's Rate-Distortion Theorem (Converse)

We prove that any $(2^{nR}, n)$ code achieving expected distortion $\leq D$ must have $R \geq R(D)$.

Let $M = 2^{nR}$ codewords $\hat{\mathbf{x}}(1), \ldots, \hat{\mathbf{x}}(M) \in \hat{\mathcal{X}}^n$. The encoder maps source sequences to indices; the decoder maps indices to reconstructions. The end-to-end mapping defines a conditional distribution $p(\hat{x}^n | x^n)$.

By the data-processing inequality applied to $X_1^n \to W \to \hat{X}_1^n$ (where W is the codeword index):

$$I(X_1^n; \hat{X}_1^n) \leq I(X_1^n; W) \leq \log_2 M = nR.$$

For a memoryless source $p(x_1^n) = \prod_t p(x_t)$, the mutual information satisfies

$$I(X_1^n; \hat{X}_1^n) = \sum_{t=1}^{n} I(X_t; \hat{X}_t | X_1^{t-1}, \hat{X}_{t+1}^n) \geq \sum_{t=1}^{n} I(X_t; \hat{X}_t),$$

where the last inequality follows... actually this direction is tricky. Let us use the standard approach via a time-sharing argument.

By introducing a uniformly random time index $T \sim \mathrm{Unif}(\{1, \ldots, n\})$ independent of everything:

$$nR \geq I(X_1^n; W) \geq \sum_{t=1}^{n} I(X_t; \hat{X}_t | \text{other}) \geq n I(X_T; \hat{X}_T | T).$$

Since $(X_T, \hat{X}_T)$ has joint distribution $\frac{1}{n}\sum_t p_{X_t, \hat{X}_t}$ and the expected distortion satisfies $\mathbb{E}[d(X_T, \hat{X}_T)] = \frac{1}{n}\sum_t \mathbb{E}[d(X_t, \hat{X}_t)] \leq D$, we have by definition of $R(D)$:

$$I(X_T; \hat{X}_T | T) \geq R\Big(\mathbb{E}[d(X_T, \hat{X}_T)]\Big) \geq R(D),$$

where the first inequality uses the fact that $R(D)$ is the minimum mutual information over all joint distributions with $\mathbb{E}d \leq D$, and the second uses the convexity and monotonicity of R. Combining: $R \geq R(D)$. ■

C Linear Programming Bounds (Delsarte's Method)

The Hamming and Plotkin bounds on $A(n,d)$ can be substantially improved using a linear programming relaxation due to Delsarte (1973). The key insight is that the weight distribution of any code satisfies a set of linear constraints (via the MacWilliams transform), and one can optimize over these constraints.

Delsarte LP Bound

$A(n,d) \le A^*(n,d)$, where $A^*(n,d)$ is the optimal value of the linear program:

$$\max \sum_{j=0}^{n} f_j \quad \text{subject to} \quad f_0 = 1, \quad f_j \ge 0, \quad \sum_j f_j K_i(j) \ge 0 \text{ for all } i, \quad f_j = 0 \text{ for } j = 1, \dots, d-1.$$

Here $K_i(j) = K_i^n(j) = \sum_{s=0}^{i} (-1)^s \binom{j}{s}\binom{n-j}{i-s}$ are the *Krawtchouk polynomials*.

The Delsarte LP bound gives the best known bounds on $A(n,d)$ for many parameter ranges, and sometimes gives exact values. For example, it proves that no perfect binary code with $d = 5$ exists for $n > 23$ (besides the $[23, 12, 7]$ Golay code), resolving a long-standing open problem.

D Turbo Code Distance Properties and the Error Floor

Turbo codes exhibit a distinctive BER curve: a steep *waterfall* region at moderate SNR where BER decreases rapidly, followed by a shallower *error floor* at high SNR. Understanding this floor requires analyzing the low-weight codewords of the turbo code, which depend critically on the interleaver.

Transfer function and effective free distance. The distance properties of a convolutional code are characterized by its *weight enumerator* (transfer function) $T(W, N, J)$, where $W^d N^w J^j$ indicates codewords of total weight d produced by input sequences of weight w and length j.

For turbo codes, the *effective free distance* d_{free} is the minimum weight of any codeword generated by a weight-2 input (the interleaver can spread weight-1 inputs but weight-2 inputs can produce low-weight outputs from both component codes). For the 3GPP LTE/5G turbo code with QPP interleaver of size k, $d_{\text{free}} \approx \sqrt{k}$, which grows with block length. This is why turbo codes in LTE have low error floors at the block lengths of interest.

Union bound on BER. The ML BER on the AWGN channel (BPSK, E_b/N_0 in dB) is bounded above by

$$P_b \le \sum_{d=d_{\text{free}}}^{\infty} \frac{A_d}{k} Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right),$$

where A_d is the total input weight for codewords of weight d, k is the information block length, and $Q(\cdot)$ is the complementary Gaussian CDF. At high SNR, the d_{free} term dominates, giving $P_b \approx (A_{d_{\text{free}}}/k) Q(\sqrt{2d_{\text{free}} R E_b/N_0})$.

E Connections to Cryptography and Information Security

Information-theoretic ideas underpin many foundational results in cryptography and security, forming a bridge between the compression / coding perspective and the security perspective.

Shannon's perfect secrecy. A cipher $(K, \mathcal{E}, \mathcal{D})$ achieves *perfect secrecy* if the ciphertext $C = \mathcal{E}_K(M)$ reveals no information about the plaintext M: $I(M;C) = 0$, i.e. $M \perp C$. Shannon (1949) proved: perfect secrecy requires $H(K) \ge H(M)$, i.e. the key must have at least as much entropy as the message. The one-time pad achieves perfect secrecy with $H(K) = H(M)$: $C = M \oplus K$ with $K \sim$ Unif.

Min-entropy extractors and randomness. Physical random number generators produce bits with imperfect distributions: the min-entropy $H_\infty(X) = -\log \max_x p(x)$ may be less than 1 per bit (see Volume I, Section 1.6). A *randomness extractor* (Nisan-Wigderson, 1994) takes a source X of min-entropy k and a short uniform seed and produces near-uniform output of length approaching k. Coding theory enters via the *leftover hash lemma*: a universal hash function compresses X to 2-universal near-uniform random bits, with error bounded in terms of the collision probability (Rényi entropy H_2).

Wiretap channels and physical-layer security. Wyner (1975) introduced the *wiretap channel*: Alice transmits to Bob over a main channel, while Eve eavesdrops via a degraded channel. The *secrecy capacity* is

$$C_s = \max_{p(x)} [I(X;Y) - I(X;Z)],$$

where Y is Bob's output and Z is Eve's. This is achievable using specially designed wiretap codes that combine classical channel codes with randomization to confuse Eve while enabling Bob to decode. The secrecy capacity is the maximum reliable communication rate to Bob that simultaneously keeps Eve in the dark, and is achievable without any shared secret key.

F Open Problems in Coding Theory and Compression

1. The GV gap. Does there exist a binary linear code family achieving rate $> 1 - h_b(\delta)$ with relative distance δ? All known binary code constructions meet (but do not beat) the GV bound. Algebraic-geometry codes beat GV for $q \to \infty$ but not for $q = 2$.

2. Efficient LDPC codes with large minimum distance. LDPC codes approach capacity via iterative decoding, but their minimum distances grow only sub-linearly with block length. Constructing LDPC codes that simultaneously have (a) BP threshold near capacity, (b) linearly growing minimum distance, and (c) efficient encoding, remains open.

3. Polar codes: speed of polarization. Polar codes (Arikan, 2009) provably achieve capacity for binary symmetric channels, but the block error probability decays as $O(2^{-n^\beta})$ for $\beta < 1/2$. Whether this can be improved to $O(2^{-n^{1/2+\epsilon}})$ for some $\epsilon > 0$ is open.

4. Rate-distortion for non-i.i.d. sources. The operational rate-distortion function for non-stationary or non-ergodic sources is not fully understood. In particular, computing $R(D)$ for Markov sources with general distortion measures requires efficient algorithms beyond Blahut-Arimoto.

5. Universal lossy compression. While LZ-type algorithms are universal for lossless compression (universally achieving entropy rate), the analogous problem for lossy compression is harder: does there exist a universal lossy compressor approaching $R(D)$ for all stationary ergodic sources? Partial results (Linder et al.) are known for specific distortion measures, but a fully general solution is open.

6. Algorithmic rate-distortion. Define the *algorithmic rate-distortion function* $R_K(D)$ using Kolmogorov complexity as the natural analogue of $R(D)$: the minimum algorithmic information about x needed to reconstruct x to within distortion D. The relation between $R_K(D)$ and $R(D)$ is known only to within $O(\log n)$ additive terms; closing this gap would unify the algorithmic and information-theoretic approaches to lossy compression.

Cross-Volume Connections and Methodological Synthesis

From Volume I. The channel coding theorem (Volume I, Part 4) establishes that capacity is achievable but does not construct achieving codes. Volume II fills this gap: LDPC codes and turbo codes approach capacity with efficient decoding. The AEP (Volume I, Part 3) underlies the achievability proof for rate-distortion; the method of types (Volume I, Section 3.6) gives a cleaner proof of the converse. The exponential family framework (Volume I, Section 2.6) is the algebraic structure underlying the Blahut-Arimoto algorithm: the fixed-point equations are the same as those for the log-partition function of an exponential family.

Toward Volume III. The Fisher information metric on the space of probability distributions (Volume III) is connected to coding theory via the *algebraic geometry codes*: Reed-Solomon codes are the $q \to \infty$ limit of Goppa codes, which are defined using algebraic curves over $\mathbb{F}_q$ and whose minimum distances are determined by the genus of the curve. The dual of a q-ary linear code is a statistical model in the exponential family sense; the MacWilliams identity is a Fourier duality on the group $\mathbb{F}_q^n$ that generalizes to the Hadamard transform on exponential families. Density evolution for LDPC codes can be reinterpreted as a dynamical system on the space of probability distributions, with the threshold corresponding to a bifurcation in the dynamics.

The *natural gradient* (Amari, Volume III) arises naturally in LDPC decoding: the sum-product algorithm is an instance of expectation-maximization in a graphical model, and the natural gradient direction for optimizing the degree distribution is the Fisher-efficient update. These connections are developed fully in Volume III.

The unified picture. The three volumes of *Mathematics of Information & Entropy* together cover the following landscape:

- Volume I: *Measures of information* — entropy, divergence, capacity — and their operational meanings via coding theorems (lossless and noisy).
- Volume II: *Constructive codes and algorithms* — how to build codes achieving the limits of Volume I, and how to compress sources with and without distortion.
- Volume III: *Geometry of information* — the Riemannian structure on probability distributions, the natural gradient, statistical manifolds, and the exponential family as the geometry underlying all of the above.

The thread running through all three volumes is the interplay between *probability*, *algebra*, and *geometry*, with information always playing the central organizing role.

Extended Case Study: End-to-End Design of a Data Storage System

This case study integrates the major themes of Volume II into a single, concrete engineering problem: the design of an error-correction and compression system for a solid-state drive (SSD) storing a corpus of text data. We will combine RS outer codes, LDPC inner codes, arithmetic compression, and rate-distortion analysis into a coherent system design.

System architecture. An SSD stores data in flash memory cells, each of which can represent 2 bits (MLC) or 3 bits (TLC) in its charge level. Flash memory has two main error mechanisms:

1. *Random bit errors* (from charge leakage): modeled as a BSC with crossover probability $\epsilon \approx 10^{-3}$ (raw BER).
2. *Burst errors* (from program/erase operations): entire pages (typically 4–16 KB) can fail.

The solution is a *concatenated code*: an inner LDPC code corrects random bit errors within each page, and an outer Reed-Solomon code corrects burst page failures across the storage array.

Inner code: LDPC design. Specifications: block length $n = 4096$ bits (one SSD page), target post-LDPC BER $\leq 10^{-12}$, raw channel BER $\epsilon_{\text{raw}} = 10^{-3}$ (BSC model). We choose a $(3, 6)$-regular LDPC code of rate $R = 1/2$, giving $k = 2048$ information bits per page.

BP threshold for $(3, 6)$-regular LDPC on BSC: $\epsilon^* \approx 0.084$ (by density evolution on BSC). Since $\epsilon_{\text{raw}} = 10^{-3} \ll 0.084$, we operate in the waterfall region and BP converges in < 10 iterations with negligible error.

Post-LDPC BER: numerical density evolution gives post-decoding BER $\approx 10^{-15}$ at $\epsilon_{\text{raw}} = 10^{-3}$ with $T = 20$ iterations. Excellent margin.

Outer code: Reed-Solomon. Specifications: correct up to $t_{\text{RS}} = 2$ page failures per stripe of $n_{\text{RS}} = 32$ pages; rate ≥ 0.9.

Choose RS$(32, 28, 5)$ over $\mathbb{F}_{2^8}$: $n = 32$, $k = 28$, $d = 5$, corrects $t = 2$ page failures. Rate $= 28/32 = 0.875$. Total system rate (inner $\times$ outer): $0.5 \times 0.875 = 0.4375$ information bits per raw stored bit.

Compression layer: arithmetic coding. The text data to be stored has been measured to have empirical entropy $H \approx 4.5$ bits/ASCII character (accounting for English redundancy). Each character is 8 bits, so the compression ratio achievable is $H/8 \approx 0.5625$. Arithmetic coding with an adaptive model achieves this ratio within 0.01 bits/character overhead.

Overall information density: arithmetic-coded data at rate 0.5625, then stored with FEC overhead at rate 0.4375:

$$\text{Information bits per raw bit} = 0.5625 \times 0.4375 \approx 0.246.$$

Equivalently, the system stores approximately 0.246 bits of original text information per raw flash bit, with reliability $\geq 1 - 10^{-15}$ per bit.

Rate-distortion consideration for compressed images. SSDs also store compressed images (JPEG, WebP). JPEG uses DCT coefficients quantized according to a rate-distortion curve; the overall system operates at roughly $R = 0.5$ bits/pixel with PSNR ≈ 35 dB for natural images. The Gaussian rate-distortion bound $R(D) = \frac{1}{2}\log_2(\sigma^2/D)$ predicts an information-theoretic lower bound of approximately 0.4 bits/pixel for the same quality level (assuming i.i.d. Gaussian pixel model), so JPEG operates within 0.1 bits/pixel of the theoretical opti-

mum.

System-level summary.

Layer	Method	Parameters
Source compression	Arithmetic coding	$H \approx 4.5$ bits/char, overhead < 0.01
Inner FEC	$(3, 6)$-regular LDPC	$[4096, 2048]$, BP threshold $\epsilon^* = 0.084$
Outer FEC	Reed-Solomon	RS$(32, 28, 5)$ over $\mathbb{F}_{256}$, corrects 2 page failures
Overall rate	Combined	0.246 info bits per raw bit
Reliability	Post-decoding BER	$< 10^{-15}$ per bit

This design illustrates how the three main themes of Volume II — algebraic codes (RS), iterative codes (LDPC), and data compression (arithmetic coding with rate-distortion analysis) — combine into a coherent, practically effective system.

Bibliographic Notes

The material in this volume draws on the following primary sources.

Part 1 (block codes and combinatorial bounds): The Hamming bound and Hamming codes are due to Hamming (1950), who also introduced the concept of Hamming distance. The Singleton bound is from Singleton (1964). The Plotkin bound appears in Plotkin (1960). The GV bound was established independently by Gilbert (1952) and Varshamov (1957). The Delsarte LP bound (Appendix C) is from Delsarte (1973).

Part 2 (linear codes and Reed-Solomon): Linear codes in systematic form were studied extensively by Slepian, Prange, and others in the 1950s. Reed-Solomon codes are from Reed and Solomon (1960). The Berlekamp-Massey algorithm is due to Berlekamp (1968) and Massey (1969); the Euclidean algorithm decoder to Sugiyama et al. (1975). BCH codes were independently discovered by Bose, Ray-Chaudhuri, and Hocquenghem (1959–1960).

Part 3 (LDPC and turbo codes): LDPC codes were introduced by Gallager in his 1963 thesis. Their rediscovery and the connection to belief propagation is due to MacKay and Neal (1996). Density evolution for the BEC was introduced by Luby et al. (2001). Turbo codes are due to Berrou, Glavieux, and Thitimajshima (1993). The BCJR algorithm is from Bahl, Cocke, Jelinek, and Raviv (1974). Irregular LDPC optimization via density evolution is due to Richardson and Urbanke (2001); the capacity-approaching LDPC threshold analysis appears in Richardson, Shokrollahi, and Urbanke (2001). Protograph LDPC codes are due to Thorpe (2003).

Part 4 (data compression): The Huffman algorithm is from Huffman (1952). Arithmetic coding was developed by Rissanen and Langdon (1979), with integer-arithmetic implementations by Pasco (1976) and Rissanen-Langdon (1981). LZ77 and LZ78 are from Ziv and Lempel (1977, 1978). LZW is due to Welch (1984). Rate-distortion theory originates with Shannon (1959). The Blahut-Arimoto rate-distortion algorithm is the companion to the channel capacity algorithm of Blahut (1972) and Arimoto (1972).

Part 5 (Kolmogorov complexity): Kolmogorov complexity was introduced independently by Kolmogorov (1965), Solomonoff (1964), and Chaitin (1966). The invariance theorem is due to Kolmogorov. The universal semi-measure and Solomonoff induction are from Solomonoff (1964). The incompressibility method was systematized by Li and Vitányi in their textbook. The MDL principle is from Rissanen (1978). Logical depth is due to Bennett (1988). The standard reference for algorithmic information theory is Li and Vitányi (2019), cited in the bibliography below.

Bibliography

1. R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

2. C. E. Shannon, "Coding Theorems for a Discrete Source with a Fidelity Criterion," in *IRE National Convention Record*, vol. 4, pp. 142–163, 1959.

3. R. C. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

4. R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, 1963.

5. R. G. Gallager, *Information Theory and Reliable Communication*, Wiley, 1968.

6. E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, 1968.

7. J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969.

8. L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

9. J. Rissanen, "Generalized Kraft Inequality and Arithmetic Coding," *IBM Journal of Research and Development*, vol. 20, no. 3, pp. 198–203, 1976.

10. R. Blahut, "Computation of Channel Capacity and Rate-Distortion Functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

11. S. Arimoto, "An Algorithm for Computing the Capacity of Arbitrary Discrete Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, 1972.

12. J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

13. J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

14. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed., Wiley, 2006.

15. D. J. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

16. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," in *Proceedings of IEEE ICC 1993*, pp. 1064–1070.

17. T. J. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.

18. M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient Erasure Correcting Codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

19. T. J. Richardson, M. A. Shokrollahi, and R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, vol. 47,

no. 2, pp. 619–637, 2001.

20. A. N. Kolmogorov, "Three Approaches to the Quantitative Definition of Information," *Problems of Information Transmission*, vol. 1, no. 1, pp. 1–7, 1965.

21. R. J. Solomonoff, "A Formal Theory of Inductive Inference," *Information and Control*, vol. 7, nos. 1–2, pp. 1–22 and 224–254, 1964.

22. M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 4th ed., Springer, 2019.

23. J. Rissanen, "Modeling by Shortest Data Description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

24. C. H. Bennett, "Logical Depth and Physical Complexity," in *The Universal Turing Machine: A Half-Century Survey*, R. Herken, ed., Oxford University Press, 1988.

25. P. Delsarte, "An Algebraic Approach to the Association Schemes of Coding Theory," *Philips Research Reports Supplements*, no. 10, 1973.

26. E. C. Welch (T. A. Welch), "A Technique for High-Performance Data Compression," *Computer*, vol. 17, no. 6, pp. 8–19, 1984.

27. A. Arikan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

28. I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*, 2nd ed., Cambridge University Press, 2011.

29. R. W. Yeung, *Information Theory and Network Coding*, Springer, 2008.

30. A. D. Wyner, "The Wire-Tap Channel," *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, 1975.

Part 6 — Polar Codes and Channel Polarization

Arikan's channel polarization phenomenon: synthetic channels from n uses of a binary-input memoryless channel. Bhattacharyya parameters and polarization bounds. Polar code construction: frozen bits and information bits. Successive cancellation decoding. Successive cancellation list decoding and CRC-aided polar codes. Finite-length analysis and comparison with LDPC and turbo codes. Polar codes in 5G NR.

6.1 The Polarization Phenomenon

Polar codes, introduced by Erdal Arıkan in 2008, are the first provably capacity-achieving family of codes with an explicit construction and efficient $O(n \log n)$ encoding and decoding algorithms. Their foundation is the *channel polarization* phenomenon: by combining $n = 2^{\ell}$ independent uses of a binary-input memoryless channel W, one synthesizes n *synthetic channels*, half of which have capacity approaching 1 (very reliable) and half approaching 0 (completely noisy), as $n \to \infty$.

Binary-Input Memoryless Channel

A *binary-input memoryless channel* (B-MC) $W : \{0,1\} \to \mathcal{Y}$ is specified by its transition probability $W(y|x)$. The *symmetric capacity* (or *Bhattacharyya-based capacity*) is

$$I(W) = \sum_{y\in\mathcal{Y}} \sum_{x\in\{0,1\}} \frac{1}{2} W(y|x) \log_2 \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)},$$

the mutual information between a uniform input $X \sim \mathrm{Unif}(\{0,1\})$ and output Y. The *Bhattacharyya parameter* of W is

$$Z(W) = \sum_{y\in\mathcal{Y}} \sqrt{W(y|0)W(y|1)}.$$

Note $Z(W) \in [0,1]$, with $Z(W) = 0$ for a perfect channel and $Z(W) = 1$ for a completely noisy channel. The bound $I(W) + Z(W) \leq 1$ holds (with near-equality for BEC and BSC), so a channel is reliably usable iff $Z(W) \approx 0$ iff $I(W) \approx 1$.

The Two-Channel Combining Operations

Start with $n = 2$ independent uses of channel W. Define two input bits $u_1, u_2 \in \{0,1\}$ and set

$$x_1 = u_1 \oplus u_2, \qquad x_2 = u_2.$$

Transmit x_1, x_2 over two independent copies of W, receiving y_1, y_2. The pair (y_1, y_2) constitutes the channel output. Two synthetic channels arise:

Channel Combining: W^- and W^+

From two independent copies of W, define:

- *Bad channel* $W^- : \{0,1\} \to \mathcal{Y}^2$: $W^-(y_1, y_2|u_1) = \sum_{u_2} \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2)$. This is the channel seen by u_1 when u_2 is marginalized.
- *Good channel* $W^+ : \{0,1\} \to \mathcal{Y}^2 \times \{0,1\}$: $W^+(y_1, y_2, u_1|u_2) = \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2)$. This is the channel seen by u_2 when u_1 is given as side information.

Polarization identities. The capacities and Bhattacharyya parameters of the synthetic channels satisfy:

$$I(W^-) + I(W^+) = 2I(W), \qquad Z(W^-)^2 + Z(W^+)^2 \leq 2Z(W)^2.$$

More explicitly for Bhattacharyya parameters:

$$Z(W^-) = 2Z(W) - Z(W)^2 \geq Z(W), \qquad Z(W^+) = Z(W)^2 \leq Z(W).$$

So W^- is *worse* than W (larger Z) and W^+ is *better* (smaller Z). The combining operation *polarizes* the channels.

Proof of the Bhattacharyya recursions. For W^+: $Z(W^+) = \sum_{y_1,y_2,u_1} \sqrt{W^+(y_1, y_2, u_1|0) W^+(y_1, y_2, u_1|1)}$. Substituting the definition and summing over u_1:

$$\begin{aligned} Z(W^+) &= \sum_{y_1,y_2} \sum_{u_1} \sqrt{\frac{1}{2}W(y_1|u_1) \cdot \frac{1}{2}W(y_2|0) \cdot \frac{1}{2}W(y_1|u_1 \oplus 1) \cdot \frac{1}{2}W(y_2|1)} \\ &= \frac{1}{2} \sum_{y_2} \sqrt{W(y_2|0)W(y_2|1)} \cdot \sum_{y_1} \sum_{u_1} \sqrt{W(y_1|u_1)W(y_1|u_1 \oplus 1)}. \end{aligned}$$

Note $\sum_{y_1}\sum_{u_1\in\{0,1\}}\sqrt{W(y_1|u_1)W(y_1|u_1\oplus 1)} = 2\sum_{y_1}\sqrt{W(y_1|0)W(y_1|1)} = 2Z(W)$. So $Z(W^+) = \frac{1}{2}\cdot Z(W)\cdot 2Z(W) = Z(W)^2$. ✓

For W^-: $Z(W^-) \le 2Z(W) - Z(W)^2$ follows from the triangle inequality for $\ell^{1/2}$ norms; with equality for the BEC.

Recursive Polarization and the Polar Code Transform

Apply the two-channel operation recursively to $n = 2^\ell$ channels. At each level, each channel splits into a $(\cdot)^-$ (bad) and $(\cdot)^+$ (good) version. After ℓ levels, we have $n = 2^\ell$ synthetic channels $W_n^{(i)}$ for $i = 1, \dots, n$.

Polarization theorem (Arıkan, 2009). For any B-MC W with $0 < I(W) < 1$, as $n = 2^\ell \to \infty$:

$$\frac{1}{n}\left|\left\{i : Z(W_n^{(i)}) < 2^{-n^\beta}\right\}\right| \to I(W) \quad \text{and} \quad \frac{1}{n}\left|\left\{i : Z(W_n^{(i)}) > 1 - 2^{-n^\beta}\right\}\right| \to 1 - I(W)$$

for any $\beta < 1/2$. That is, fraction $I(W)$ of synthetic channels become very good (reliable), and fraction $1 - I(W)$ become very bad (useless), with only a vanishing fraction remaining in between.

Example — Polarization on the BEC

For a BEC with erasure probability ϵ: $I(W) = 1-\epsilon$, $Z(W) = \epsilon$. The recursive formulas give exact results: $Z(W^-) = 2\epsilon - \epsilon^2$, $Z(W^+) = \epsilon^2$. Starting from $\epsilon = 0.5$ (capacity 0.5):

Level	$Z(W^-)$	$Z(W^+)$
0	0.500	0.500
1	0.750	0.250
2 (from W^-)	0.9375	0.5625
2 (from W^+)	0.4375	0.0625

After $\ell = 3$ levels ($n = 8$): Bhattacharyya parameters include values near 0 (reliable channels) and near 1 (useless channels), with $4 \approx n \cdot I(W)$ good channels and 4 bad channels. Polarization is clearly underway.

6.2 Polar Code Construction and Encoding

Polar Code

An (n, k) polar code for channel W is defined by:

- Block length $n = 2^\ell$.
- An *information set* $\mathcal{A} \subset \{1, \dots, n\}$ of size k, corresponding to the k synthetic channels with smallest Bhattacharyya parameters (best channels). These carry the k information bits $\mathbf{u}_\mathcal{A}$.
- A *frozen set* $\mathcal{A}^c = \{1, \dots, n\} \setminus \mathcal{A}$ of size $n - k$, where bits are set to 0 (or any fixed known value). These occupy the worst channels.

The code rate is $R = k/n$.

Encoding. Polar encoding uses the *polar transform* (also called the *generator matrix* G_n):

$$G_n = B_n F^{\otimes \ell}, \qquad F = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

where $F^{\otimes \ell}$ is the ℓ-th Kronecker power of F, and B_n is the bit-reversal permutation matrix. The codeword is $\mathbf{x} = \mathbf{u}G_n$ (where $\mathbf{u}$ contains information bits in $\mathcal{A}$ and zeros in $\mathcal{A}^c$).

The encoding can be implemented in $O(n \log n)$ operations using the butterfly structure of the polar transform, identical to the FFT butterfly.

Example — Polar code $n = 4$ over BEC(0.5)

$n = 4$, $\ell = 2$. Bhattacharyya parameters after recursive polarization from $Z = 0.5$:

Channel i	$Z(W_4^{(i)})$	Status
1	0.9375	Frozen
2	0.5625	Frozen
3	0.4375	Information
4	0.0625	Information

The $(4, 2)$ polar code uses $\mathcal{A} = \{3, 4\}$ as information bits.
$G_4 = B_4 F^{\otimes 2}$. $F^{\otimes 2} = F \otimes F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$. With B_4 = bit-reversal permutation $(1, 2, 3, 4) \to (1, 3, 2, 4)$, one obtains a specific G_4.
Information word $(u_3, u_4) = (1, 0)$: set $\mathbf{u} = (0, 0, 1, 0)$ (frozen bits $u_1 = u_2 = 0$).
Codeword $\mathbf{x} = \mathbf{u}G_4 =$ (row 3 of G_4) (only $u_3 = 1$ is nonzero).

6.3 Successive Cancellation Decoding

The natural decoder for polar codes is *successive cancellation* (SC) decoding. It decodes the synthetic bits $u_1, u_2, \ldots, u_n$ one by one, from u_1 to u_n.

For each position i:

- If $i \in \mathcal{A}^c$ (frozen): set $\hat{u}_i = 0$ (the known frozen value).
- If $i \in \mathcal{A}$ (information): compute the likelihood ratio $L_n^{(i)}(\mathbf{y}_1^n, \hat{\mathbf{u}}_1^{i-1})$ for channel $W_n^{(i)}$ given the received sequence $\mathbf{y}$ and previous decisions $\hat{u}_1, \ldots, \hat{u}_{i-1}$, then decide $\hat{u}_i = \mathbf{1}[L_n^{(i)} < 0]$.

The LLRs $L_n^{(i)}$ can be computed recursively using the same butterfly structure as encoding, in $O(n \log n)$ time total. The recursion (in LLR domain) is:

SC Decoding LLR Recursions

Let $L_1^{(1)}(y) = \log \frac{W(y|0)}{W(y|1)}$ be the channel LLR.
For level ℓ computations combining two blocks of LLRs L_a and L_b:

$$L^{(\text{upper})} = \phi(L_a, L_b) \triangleq 2 \operatorname{arctanh}(\tanh(L_a/2) \cdot \tanh(L_b/2)) \approx \operatorname{sign}(L_a) \cdot \operatorname{sign}(L_b) \cdot \min(|L_a|, |L_b|) \quad \text{(min-sum}$$

$$L^{(\text{lower})} = L_b + (1 - 2\hat{u}) L_a \quad \text{(given decision } \hat{u} \text{ for upper bit)}.$$

These are analogous to the check-node and variable-node updates in LDPC BP decoding.

Block error probability of SC decoding. The block error probability of SC decoding satisfies

$$P_{\text{block}}^{\text{SC}} \leq \sum_{i \in \mathcal{A}} Z(W_n^{(i)}),$$

by a union bound over information bit positions. Since $Z(W_n^{(i)}) \leq 2^{-n^\beta}$ for information bits in the limit, this gives $P_{\text{block}}^{\text{SC}} \leq k \cdot 2^{-n^\beta}$, which vanishes super-polynomially in n.

6.4 Successive Cancellation List Decoding

SC decoding has a key drawback: it is a greedy algorithm that cannot recover from an early incorrect decision. *Successive cancellation list* (*SCL*) *decoding* (Tal and Vardy, 2015) maintains a list of L candidate codewords simultaneously and chooses the most likely one at the end.

The list decoder. At each information bit position i, instead of making a hard decision, the decoder branches: it maintains up to L candidate paths. For each current path, it creates two new paths (with $\hat{u}_i = 0$ and $\hat{u}_i = 1$) and keeps only the L paths with the highest cumulative path metrics. If the list size L is doubled at each step, this is exponential; the key insight is that one can prune efficiently to keep exactly L paths at each step using a partial likelihood metric.

At the end of n steps, the decoder has L candidate codewords and selects the one with the highest path metric (or, in CRC-aided decoding, the highest-metric codeword that passes the CRC check).

CRC-aided polar codes (CA-SCL). Modern polar codes append a *cyclic redundancy check* (CRC) to the information bits before polar encoding. After SCL decoding, only candidate codewords that pass the CRC are considered valid, and the decoder outputs the highest-metric valid candidate.

CRC-aided polar codes achieve near-ML decoding performance and outperform turbo codes and LDPC codes at short and moderate block lengths ($k \leq 1024$ bits). This is why the 3GPP 5G NR standard selected polar codes for control channel coding (PDCCH/PUCCH) at short block lengths, while LDPC codes handle data channels at longer block lengths.

Example — Comparison at $k = 256$, rate $1/2$, AWGN

At block length $n = 512$, rate $1/2$, AWGN channel, BPSK modulation:

Code / Decoder	E_b/N_0 at BLER $= 10^{-3}$	Decoding complexity
Turbo (LTE), 8 iter.	1.8 dB	$O(n)$ per iter.
LDPC (5G NR), BP 50 iter.	1.7 dB	$O(n)$ per iter.
Polar (5G NR), SCL $L = 8$, CRC-11	1.5 dB	$O(Ln \log n)$
Polar, SCL $L = 32$, CRC-11	1.4 dB	$O(Ln \log n)$
ML lower bound (simulation)	1.1 dB	exponential

CA-SCL polar codes are closest to the ML bound at this block length.

6.5 Polar Codes in 5G NR

The 3GPP 5G NR standard (Release 15, 2017) is the first commercial wireless standard to adopt polar codes, selected for control channel coding (downlink PDCCH and uplink PUCCH) after a competitive evaluation against LDPC and turbo codes.

5G NR polar code specifications.

- Block lengths: $n \in \{64, 128, 256, 512, 1024\}$ (powers of 2).
- Rate: $R \in [1/6, 1]$, with rate matching via puncturing and shortening.
- CRC lengths: CRC-6 for small blocks, CRC-11 for larger blocks.

- Decoder: CA-SCL with list size $L = 8$ (uplink) or $L = 4$ (downlink).
- Information set $\mathcal{A}$: determined by a reliability sequence derived from density evolution on the AWGN channel, stored as a fixed table of $n = 1024$ indices sorted by reliability.

The choice of polar codes for control channels reflects their superior performance at short block lengths ($k \leq 512$ bits), where ML-approaching decoding is tractable with moderate list size L. For data channels (PDSCH/PUSCH), 5G NR uses quasi-cyclic LDPC codes (protograph-based, derived from 3GPP's "base graph" design), which achieve better throughput at large block lengths due to hardware-friendly architecture.

Part 7 — Advanced Topics in Source Coding

Distributed source coding: Slepian-Wolf theorem (lossless multi-terminal compression). The Wyner-Ziv theorem: lossy compression with side information at the decoder. Common information and the Gray-Wyner network. Successive refinement: broadcasting a source at multiple rates. Multiple description coding. Connections to network information theory and Shannon's multi-terminal coding framework.

7.1 The Slepian-Wolf Theorem

The *Slepian-Wolf theorem* (1973) is the fundamental result of multi-terminal source coding. It establishes that two correlated sources X and Y can be compressed *separately* (without communication between encoders) at rates approaching those achievable with *joint* encoding, as long as the decoder has access to both compressed streams.

Slepian-Wolf Theorem

Let (X, Y) be a pair of discrete correlated sources with joint distribution $p(x, y)$. Two separate encoders compress X and Y at rates R_X and R_Y respectively; a joint decoder recovers (X, Y) with vanishing error probability as $n \to \infty$. The set of achievable rate pairs (R_X, R_Y) is:

$$\mathcal{R}_{\text{SW}} = \{(R_X, R_Y) : R_X \geq H(X|Y),\ R_Y \geq H(Y|X),\ R_X + R_Y \geq H(X, Y)\}.$$

This is the same region achievable by *joint* encoding!

Operational interpretation. The surprising content is that knowing Y at the decoder (but not at the encoder) allows X to be compressed at rate $H(X|Y)$ rather than $H(X)$. The encoder for X does not know Y, yet manages to leverage the correlation via channel-coding-like binning.

Achievability via binning. The encoder for X assigns each x^n to one of 2^{nR_X} bins uniformly at random (a "random binning" encoder). The bin index $B_X(x^n)$ is transmitted at rate R_X. Similarly for Y with bin index $B_Y(y^n)$. The decoder, given (B_X, B_Y), looks for the unique jointly typical pair $(\hat{x}^n, \hat{y}^n)$ in the intersection of the two bins.

For decoding to succeed, we need the intersection of the bin for x^n and the bin for y^n to contain no other jointly typical pair (with high probability). The probability of a "bad" pair $(\tilde{x}^n, y^n)$ with $\tilde{x}^n \neq x^n$ landing in the same bin as x^n and being jointly typical with y^n is approximately $2^{-nR_X} \cdot 2^{-nI(X;Y)}$ per pair, and there are about $2^{nH(X|Y)}$ such pairs, so the expected number of confusable pairs is $2^{n(H(X|Y)-R_X)}$, which vanishes when $R_X > H(X|Y)$. A symmetric argument handles R_Y, and the joint rate constraint $R_X + R_Y > H(X, Y)$ handles the case where both are marginalized.

Converse. By Fano's inequality: if (R_X, R_Y) is achievable with vanishing error, then for any $\epsilon > 0$ and large n:

$$nR_X + nR_Y \geq H(B_X, B_Y) \geq H(B_X, B_Y|\text{error}) \geq H(X^n, Y^n) - n\epsilon = nH(X, Y) - n\epsilon.$$

Individual rate constraints follow similarly. ■

Example — Binary Slepian-Wolf: correlated bits

$X = Y \oplus Z$ where $Y \sim \text{Bern}(1/2)$ and $Z \sim \text{Bern}(p)$ independent (so X is Y flipped

with probability p). Then:

$$H(X) = H(Y) = 1 \text{ bit}, \quad H(X|Y) = H(Y|X) = h_b(p),$$
$$H(X,Y) = 1 + h_b(p).$$

Slepian-Wolf region: $R_X \geq h_b(p)$, $R_Y \geq h_b(p)$, $R_X + R_Y \geq 1 + h_b(p)$. For $p = 0.1$: $h_b(0.1) = 0.469$ bits. Compare to independent compression: $R_X + R_Y \geq H(X) + H(Y) = 2$ bits. Slepian-Wolf requires only 1.469 bits total — a savings of 0.531 bits/pair from exploiting correlation.

7.2 The Wyner-Ziv Theorem: Lossy Compression with Side Information

The *Wyner-Ziv theorem* (1976) extends Slepian-Wolf to the lossy setting: X is to be compressed at minimum rate R such that the decoder, given compressed X and side information Y (correlated with X), can reconstruct $\hat{X}$ with expected distortion $\leq D$.

Wyner-Ziv Rate-Distortion Function

For source X, side information Y at the decoder (not at the encoder), and distortion measure $d(x, \hat{x})$, the minimum rate achievable is

$$R^{WZ}(D) = \min_{p(\hat{x}|x,y),\, \mathbf{u}:\ \hat{X}-U-X-Y,\ \mathbb{E}d(X,\hat{X})\leq D} [I(X;U|Y) - I(U;Y)],$$

which simplifies for many standard sources to

$$R^{WZ}(D) = \min_{p(u|x):\ \hat{x}=f(u,y),\ \mathbb{E}d(X,\hat{X})\leq D} I(X;U|Y).$$

The surprise: no penalty for not knowing Y at the encoder. When the source and side information are jointly Gaussian — $X \sim \mathcal{N}(0, \sigma^2)$ and $Y = X + N$ with $N \sim \mathcal{N}(0, N_0)$ independent — one can show that

$$R^{WZ}(D) = R(D) = \frac{1}{2}\log_2 \frac{\sigma^2_{X|Y}}{D} = \frac{1}{2}\log_2 \frac{\sigma^2 N_0/(\sigma^2 + N_0)}{D},$$

where $\sigma^2_{X|Y}$ is the conditional variance of X given Y. That is, the Wyner-Ziv rate equals the rate-distortion function of X given Y — as if the encoder also knew Y. This is the *Wyner-Ziv source coding without loss* (in terms of rate) for the Gaussian source.

However, for general sources and distortion measures, $R^{WZ}(D) > R(D|Y)$ may hold (a positive rate penalty), and the characterization of when this penalty is zero is an open problem.

Achievability via nested codes. The Wyner-Ziv achievability scheme uses *nested codebooks*: the Slepian-Wolf idea (random binning) for the channel from X^n to codewords, combined with rate-distortion covering for the lossy part. The encoder maps x^n to a codeword u^n in a covering codebook (covering the space of x^n sequences), then transmits the *bin index* of u^n in a nested binning structure. The decoder, given the bin index and y^n, finds the jointly typical u^n in the bin and computes $\hat{x}^n = f(u^n, y^n)$.

7.3 Successive Refinement

Successive refinement addresses the following problem: a source X is to be described at two rates $R_1 < R_2$, producing two descriptions B_1 (of rate R_1) and B_2 (of rate $R_2 - R_1$). A

decoder with only B_1 achieves distortion D_1; a decoder with both B_1 and B_2 achieves distortion $D_2 < D_1$. The question is: when can two operating points (R_1, D_1) and (R_2, D_2) on the rate-distortion curve be achieved *simultaneously* with nested descriptions?

Successive Refinability

A source X with distortion measure d is *successively refinable* at points (R_1, D_1) and (R_2, D_2) if there exists a coding scheme producing descriptions B_1 (rate R_1) and B_2 (rate $R_2 - R_1$) such that:

- Decoder 1 (from B_1 only) achieves distortion D_1.
- Decoder 2 (from B_1, B_2) achieves distortion D_2.

A source is *universally successively refinable* if this holds for all pairs (R_1, D_1) and (R_2, D_2) on the rate-distortion curve.

Characterization (Equitz and Cover, 1991). A source is successively refinable at $(R_1, D_1), (R_2, D_2)$ if and only if the following Markov chain holds at the optimizing distributions: $X - \hat{X}_2 - \hat{X}_1$, where $\hat{X}_1, \hat{X}_2$ are the optimal reconstructions for D_1 and D_2 respectively. In words: the coarser reconstruction $\hat{X}_1$ is a function of the finer reconstruction $\hat{X}_2$, not directly of X.

Gaussian source is universally successively refinable. For $X \sim \mathcal{N}(0, \sigma^2)$ with squared-error distortion, the optimal reconstruction at level D is $\hat{X}(D) = \sigma^2/(\sigma^2 + D) \cdot (X + Z_D)$ for an appropriate noise Z_D, and the Markov chain $X - \hat{X}_2 - \hat{X}_1$ holds because $\hat{X}_1$ is a further noisy version of $\hat{X}_2$. Hence Gaussian sources are universally successively refinable, and JPEG 2000's wavelet decomposition is a practical approximation of this property.

7.4 Multiple Description Coding

Multiple description (MD) coding addresses robustness to packet loss: a source is encoded into L descriptions, each transmitted over an independent unreliable channel. Any subset of the received descriptions must allow reconstruction with some quality, where more received descriptions give better quality.

The two-description problem (El Gamal and Cover, 1982) seeks the region of achievable $(R_1, R_2, D_0, D_1, D_2)$ triples, where R_i is the rate of description i, D_0 is the distortion when *both* descriptions are received, and D_i is the distortion when only description i is received.

The known inner bound (El Gamal-Cover) and the optimal characterization for Gaussian sources (Ozarow, 1980) show that there is a fundamental tradeoff between side distortions D_1, D_2 and central distortion D_0: improving D_0 (central quality) forces either D_1 or D_2 (or both) to worsen. The optimal tradeoff for Gaussian sources satisfies

$$2^{-2R_{\text{total}}} \leq D_0 \leq D_1 D_2 / \sigma^2,$$

where $R_{\text{total}} = R_1 + R_2$ is the total rate, reflecting the tension between redundancy (for robustness) and compression efficiency.

Part 8 — Advanced Channel Coding Topics

Capacity of the AWGN channel: power constraint, Gaussian capacity formula, the water-filling solution for parallel channels. The broadcast channel and multiple-access channel: capacity regions. The relay channel and cooperative communication. Feedback capacity. Network coding and the max-flow min-cut theorem for information flow. Fountain codes: LT codes and Raptor codes.

8.1 The AWGN Channel and Gaussian Capacity

The *additive white Gaussian noise* (*AWGN*) *channel* is the most important continuous channel model:

$$Y = X + Z, \qquad Z \sim \mathcal{N}(0, N), \quad Z \perp X,$$

with the input power constraint $\mathbb{E}[X^2] \le P$.

Shannon-Hartley Theorem

The capacity of the AWGN channel with signal power P and noise power N is

$$C = \frac{1}{2}\log_2\left(1 + \frac{P}{N}\right) \text{ bits per channel use} = W\log_2\left(1 + \frac{P}{N_0 W}\right) \text{ bits per second},$$

where the second form uses bandwidth W Hz and noise spectral density $N_0/2$ W/Hz.

Proof (achievability). Use random Gaussian codebooks: each of the 2^{nC} codewords has i.i.d. $\mathcal{N}(0, P)$ components. The received vector $\mathbf{y} = \mathbf{x} + \mathbf{z}$ is decoded by maximum likelihood, which is equivalent to nearest-neighbor decoding (choose the codeword closest in Euclidean distance to $\mathbf{y}$). By the AEP for Gaussian sources, $\|\mathbf{y} - \mathbf{x}\|^2 \approx nN$, so the "noise ball" around each codeword has radius $\sqrt{nN}$. The sphere of valid codewords (power constraint $\|\mathbf{x}\|^2 \le nP$) has radius $\sqrt{n(P+N)}$. The maximum number of non-overlapping noise balls that fit inside the signal sphere is roughly $(P/N + 1)^{n/2} = 2^{nC}$. $\blacksquare$

Proof (converse). By Fano's inequality and the data-processing inequality:

$$\begin{aligned} nR &\le I(X^n; Y^n) + n\epsilon_n \\ &= h(Y^n) - h(Y^n \mid X^n) + n\epsilon_n \\ &= h(Y^n) - h(Z^n) + n\epsilon_n \\ &\le \sum_{t=1}^{n} h(Y_t) - n \cdot \frac{1}{2}\log(2\pi e N) + n\epsilon_n \\ &\le n \cdot \frac{1}{2}\log(2\pi e(P+N)) - n \cdot \frac{1}{2}\log(2\pi e N) + n\epsilon_n \\ &= n \cdot \frac{1}{2}\log\left(1 + \frac{P}{N}\right) + n\epsilon_n, \end{aligned}$$

where the third line uses the chain rule for Gaussian; the fourth line uses $h(Y_t) \le \frac{1}{2}\log(2\pi e\mathbb{E}[Y_t^2]) \le \frac{1}{2}\log(2\pi e(P+N))$ (Gaussian maximizes entropy for fixed variance). $\blacksquare$

8.2 Water-Filling: Capacity of Parallel Gaussian Channels

Consider k independent parallel channels (e.g., subcarriers in OFDM):

$$Y_j = X_j + Z_j, \qquad Z_j \sim \mathcal{N}(0, N_j), \quad j = 1, \dots, k,$$

with total power constraint $\sum_j \mathbb{E}[X_j^2] \le P$. Maximize total capacity $C = \sum_j \frac{1}{2}\log_2(1 + P_j/N_j)$ over power allocation $\{P_j\}$, $\sum_j P_j \le P$, $P_j \ge 0$.

Water-Filling Solution

The capacity-achieving power allocation is

$$P_j^* = (\mu - N_j)^+, \qquad \text{where } (a)^+ = \max(a, 0),$$

and μ (the "water level") is chosen so that $\sum_j P_j^* = P$: $\sum_j (\mu - N_j)^+ = P$.
The corresponding total capacity is

$$C^* = \sum_{j=1}^{k} \frac{1}{2} \log_2 \left(\frac{\mu}{N_j} \right)^+ .$$

Derivation. By Lagrangian optimization: maximize $\sum_j \frac{1}{2}\log_2(1+P_j/N_j)$ subject to $\sum_j P_j = P$ and $P_j \ge 0$. The Lagrangian is $L = \sum_j \frac{1}{2}\log(1 + P_j/N_j) - \lambda(\sum_j P_j - P) + \sum_j \mu_j P_j$. KKT condition: $\frac{1}{2(N_j+P_j)} = \lambda - \mu_j$, with $\mu_j \ge 0$ and $\mu_j P_j = 0$. At the optimum: if $P_j > 0$, then $\mu_j = 0$ and $P_j = 1/(2\lambda) - N_j = \mu - N_j$; if $P_j = 0$, then $\mu_j \ge 0$. Setting $\mu = 1/(2\lambda)$ gives $P_j^* = (\mu - N_j)^+$.

Intuition. Water-filling allocates power to channels like water filling up basins of different depths (N_j). Channels that are too noisy ($N_j > \mu$) receive no power. This is the optimal strategy in OFDM, DSL, and multi-carrier wireless systems.

Example — Water-filling with $k = 3$ channels

$N_1 = 1$, $N_2 = 2$, $N_3 = 5$, total power $P = 6$.
Find μ such that $\sum_j (\mu - N_j)^+ = 6$. Try all three active: $(\mu - 1) + (\mu - 2) + (\mu - 5) = 3\mu - 8 = 6 \Rightarrow \mu = 14/3 \approx 4.67$. Check: all $N_j < \mu$? $N_3 = 5 > 4.67$: no. Channel 3 gets no power. Try two active ($j = 1, 2$): $(\mu - 1) + (\mu - 2) = 2\mu - 3 = 6 \Rightarrow \mu = 4.5$. Check: $N_3 = 5 > 4.5$. ✓ Channel 3 shut off.
Optimal allocation: $P_1^* = 3.5$, $P_2^* = 2.5$, $P_3^* = 0$. Total capacity: $C^* = \frac{1}{2}\log_2(1 + 3.5) + \frac{1}{2}\log_2(1 + 1.25) + 0 = \frac{1}{2}\log_2(4.5) + \frac{1}{2}\log_2(2.25) = \frac{1}{2}(2.170 + 1.170) = 1.670$ bits/channel use.

8.3 The Multiple-Access Channel

The *multiple-access channel* (MAC) models multiple transmitters sending to a single receiver (e.g., multiple users transmitting to a base station).

Multiple-Access Channel

A *two-user MAC* is specified by $p(y|x_1, x_2)$. Encoders 1 and 2 choose codewords X_1^n (rate R_1) and X_2^n (rate R_2) independently. The decoder, given Y^n, recovers both messages. The *capacity region* is the set of all achievable rate pairs.
For the MAC, the capacity region is:

$$\mathcal{R}_{\text{MAC}} = \bigcup_{p(x_1)p(x_2)} \{(R_1, R_2) : R_1 \le I(X_1; Y|X_2),\ R_2 \le I(X_2; Y|X_1),\ R_1 + R_2 \le I(X_1, X_2; Y)\}.$$

Achievability: successive interference cancellation. Each rate triple on the boundary of

$\mathcal{R}_{\text{MAC}}$ is achieved by a specific decoding order. Consider decoding user 1 first (treating user 2 as noise), then decoding user 2 given the decoded x_1^n. This achieves rates:

$$R_2 = I(X_2;Y) - \text{noise from user 1} \leq I(X_2;Y|X_1) \quad \text{(if user 1 decoded first)}.$$

The sum rate constraint $R_1 + R_2 \leq I(X_1, X_2; Y)$ is fundamental and cannot be exceeded regardless of decoding order. Time-sharing between different decoding orders achieves the full pentagon-shaped capacity region.

Gaussian MAC capacity region. For the Gaussian MAC $Y = X_1 + X_2 + Z$ with $Z \sim \mathcal{N}(0, N)$, $\mathbb{E}X_j^2 \leq P_j$:

$$R_1 \leq \frac{1}{2}\log_2\left(1 + \frac{P_1}{N}\right), \quad R_2 \leq \frac{1}{2}\log_2\left(1 + \frac{P_2}{N}\right), \quad R_1 + R_2 \leq \frac{1}{2}\log_2\left(1 + \frac{P_1 + P_2}{N}\right).$$

8.4 Fountain Codes: LT Codes and Raptor Codes

Fountain codes (or *rateless codes*) produce a potentially unlimited stream of encoded symbols from k source symbols, such that any $k(1+\epsilon)$ encoded symbols suffice to recover the original k symbols with high probability, for small overhead ϵ. They are ideal for broadcast and multicast channels where different receivers have different channel conditions (different loss rates).

LT Code (Luby Transform Code)

An LT code over GF(2) encodes k source bits $(s_1, \ldots, s_k)$ into an unlimited stream of *check symbols* $c_1, c_2, \ldots$, where each check symbol c_t is generated as follows:

1. Sample a degree d_t from a *degree distribution* $\Omega(d) = P[\text{degree} = d]$.
2. Choose d_t source bits uniformly at random (without replacement) as *neighbors*.
3. Set $c_t = \bigoplus_{j \in \mathcal{N}(t)} s_j$ (XOR of the selected source bits).

Decoding uses iterative (BP-like) peeling: at each step, find a degree-1 check node, recover its source bit, and remove it from all other check nodes.

The ideal soliton distribution. The degree distribution that allows the peeling decoder to succeed with high probability is the *ideal soliton distribution*:

$$\Omega(1) = \frac{1}{k}, \qquad \Omega(d) = \frac{1}{d(d-1)} \text{ for } d = 2, 3, \ldots, k.$$

This distribution ensures that at every step of the peeling process, there is exactly one degree-1 check node in expectation. However, the ideal soliton has poor concentration, leading to decoding failure with constant probability.

The *robust soliton distribution* (Luby, 2002) adds a "spike" at a particular degree $\sim \sqrt{k}\log k$ that prevents the decoder from stalling, giving decoding failure probability $O(1/k)$ from $k + O(\sqrt{k}\log^2 k)$ check symbols.

Raptor codes. *Raptor codes* (Shokrollahi, 2006) pre-code the k source bits with a high-rate outer code (typically an LDPC or LDGM code) to produce k' intermediate symbols, then apply an LT code to the intermediates. This achieves near-optimal decoding with $O(k)$ XOR operations (linear time), making them practical for large k.

Raptor codes are standardized in 3GPP for broadcast (MBMS), IETF RFC 5053 (Raptor) and RFC 6330 (RaptorQ), and used in DVB-H and DVB-IPTV.

Example — LT code with $k = 4$ source bits

Source: s_1, s_2, s_3, s_4. Suppose receiver gets check symbols: $c_1 = s_2$ (degree 1), $c_2 = s_1 \oplus s_3$ (degree 2), $c_3 = s_1 \oplus s_2 \oplus s_4$ (degree 3), $c_4 = s_3 \oplus s_4$ (degree 2).
Peeling:

1. c_1 has degree 1: $s_2 = c_1$. Remove s_2 from all: $c_3 \to s_1 \oplus s_4$ (degree 2), others unchanged.
2. $c_2 = s_1 \oplus s_3$ (degree 2): no degree-1 check. No progress yet.
3. $c_4 = s_3 \oplus s_4$ (degree 2). And $c_3 = s_1 \oplus s_4$ (degree 2). Still no degree-1.
4. We're stuck! Need another check symbol or the decoder fails.

In practice, with enough check symbols ($\approx k \log k / \log \log k$ for ideal soliton), peeling succeeds with high probability.

8.5 Network Coding and Information Flow

Classical routing in communication networks simply forwards packets: each node transmits received packets toward the destination without modification. *Network coding* (Ahlswede et al., 2000) allows intermediate nodes to *combine* (encode) multiple incoming packets before forwarding. This fundamentally changes what information flow rates are achievable.

Max-Flow Min-Cut for Information

In a directed acyclic network with m sources and a single sink (multicast), the maximum achievable rate from source s to sink t equals the *min-cut* capacity: the minimum total capacity of a set of edges whose removal disconnects s from t.

The butterfly network. The canonical example is the butterfly network with two sources s_1, s_2, two sinks t_1, t_2, and a bottleneck link of capacity 1 bit. Each sink needs both source messages. With classical routing, only 1 bit per unit time can traverse the bottleneck, limiting throughput. With network coding: the intermediate node XORs the two incoming bits and forwards the XOR, allowing each sink to recover both bits. Network coding doubles the achievable rate at each sink.

Linear network coding. The key result is that *linear* network coding (each node computes a linear combination of incoming packets over $\mathbb{F}_q$) suffices to achieve the min-cut bound, for sufficiently large field size q. *Random linear network coding* (Fragouli et al., Ho et al., 2003) chooses random linear combinations at each node; with high probability over the choice of coefficients, the min-cut is achieved simultaneously to all sinks.

Part 9 — Finite Blocklength Information Theory

The gap between Shannon limits and finite-length performance. The normal approximation: Polyanskiy-Poor-Verdú (PPV) framework. The channel dispersion. Second-order coding theorems for the BEC, BSC, and AWGN. Finite-length bounds for source coding: the Kontoyiannis-Rissanen theorem. Non-asymptotic information measures: ϵ-capacity and ϵ-achievability. Applications to short-packet communications in IoT and 5G.

9.1 Beyond Asymptotic Capacity: The PPV Framework

Classical Shannon theory is *asymptotic*: it characterizes what is achievable as block length $n \to \infty$, but says nothing about the performance at finite n. In practice, all codes have finite block length, and the gap between the Shannon limit and achievable performance at finite n matters enormously — especially for short-packet communications (IoT, control systems, 5G URLLC) where $n \leq 300$ bits.

The *Polyanskiy-Poor-Verdú (PPV) framework* (2010) provides tight second-order (i.e., $O(1/\sqrt{n})$) characterizations of finite-length performance.

Second-Order Coding Theorem (Normal Approximation)

For a channel W with capacity C and *channel dispersion* V (defined below), the maximum achievable code rate $R^*(n, \epsilon)$ at block length n and block error probability ϵ satisfies:

$$R^*(n, \epsilon) = C - \sqrt{\frac{V}{n}}\, Q^{-1}(\epsilon) + O\left(\frac{\log n}{n}\right),$$

where $Q^{-1}(\epsilon) = \sqrt{2}\,\mathrm{erfinv}(1 - 2\epsilon)$ is the inverse Q-function.

Channel Dispersion

The *channel dispersion* V of a channel W is the variance of the *information density* $\imath(X;Y) = \log \frac{W(Y|X)}{P_Y(Y)}$ under the capacity-achieving input distribution:

$$V = \mathrm{Var}_{P^*}[\imath(X;Y)] = \mathbb{E}_{P^*}\left[\left(\log \frac{W(Y|X)}{P_Y(Y)} - C\right)^2\right].$$

Examples of channel dispersion.

- *BEC*(ϵ): $C = 1 - \epsilon$, $V = \epsilon(1-\epsilon)\ln^2 2^{-1}$... more precisely $V = \epsilon(1-\epsilon)$ (nats^2). The information density takes values 0 (if erased: $\imath = 0 - (-C/(1-\epsilon))\cdot$ ...) — let us state the result: $V_{\text{BEC}} = \epsilon(1-\epsilon)(\log_2 \frac{1}{1-\epsilon})^2$.
- *BSC*(p): $C = 1 - h_b(p)$, $V_{\text{BSC}} = p(1-p)(\log_2 \frac{1-p}{p})^2$.
- *AWGN* (*SNR* = P/N): $C = \frac{1}{2}\log_2(1+P/N)$, and the dispersion is $V_{\text{AWGN}} = \frac{1}{2}\left(1 - \frac{1}{(1+P/N)^2}\right)(\log_2 e)^2$. For high SNR: $V_{\text{AWGN}} \approx \frac{(\log_2 e)^2}{2}$ nats^2/channel use.

Interpretation. The term $\sqrt{V/n}\,Q^{-1}(\epsilon)$ is the *back-off from capacity* at finite block length. For typical parameters ($\epsilon = 10^{-3}$, $Q^{-1}(10^{-3}) \approx 3.09$):

Channel	n	Back-off $\approx \sqrt{V/n} \cdot 3.09$
AWGN (0 dB SNR)	100	0.22 bits/channel use
AWGN (0 dB SNR)	1000	0.07 bits/channel use
BEC(0.5)	100	0.16 bits/channel use
BEC(0.5)	1000	0.05 bits/channel use

The back-off is significant at $n = 100$ and becomes negligible only around $n \sim 10000$. This explains why short-packet systems need carefully optimized codes and cannot rely on capacity arguments alone.

9.2 Achievability and Converse Bounds in the PPV Framework

The PPV framework provides both achievability (upper bound on P_e for a given code rate) and converse (lower bound on P_e) using hypothesis testing.

Random Coding Union Bound (PPV achievability)

For channel W^n (product of n independent uses), the minimum block error probability satisfies:

$$\epsilon^*(n, M) \leq \inf_{P_X} \mathbb{E}\left[\exp\left(-\left[\log \frac{W^n(Y^n|X^n)}{Q_{Y^n}(Y^n)} - \log M\right]^+\right)\right],$$

where Q_{Y^n} is any output distribution. Choosing $Q = P_{Y^n}$ (marginal of W^n under P_X) and $M = 2^{nR}$ gives the *random coding union* (RCU) *bound*.

Meta-Converse (PPV converse)

For any (n, M) code:

$$\epsilon \geq \max_{Q_Y} \beta_{1-1/M}(W^n(Y^n|\cdot), Q_Y^n),$$

where $\beta_\alpha(P, Q)$ is the minimum type-II error of a hypothesis test between P and Q with type-I error $\leq 1 - \alpha$.

These bounds are typically evaluated numerically and provide tight guarantees on finite-length performance. The gap between RCU achievability and meta-converse is $O(\log n/n)$, confirming the normal approximation up to this order.

9.3 Short-Packet Communications in 5G URLLC

5G *Ultra-Reliable Low-Latency Communication* (URLLC) targets block error rates of 10^{-5} with end-to-end latency ≤ 1 ms. Given 5G NR slot duration (0.5–0.125 ms) and overhead, useful payload sizes are $n_{\text{info}} \sim 20$–256 bits, with total block lengths $n \sim 64$–1024 bits.

At these parameters, the PPV normal approximation shows the required E_b/N_0 is significantly above the Shannon limit:

k (bits)	n (bits)	ϵ	C (bits/cu)	$R^*(n, \epsilon)$ / C (%)
100	200	10^{-5}	0.5	72%
100	400	10^{-5}	0.5	82%
500	1000	10^{-5}	0.5	90%

At $k = 100$, $n = 200$: operating at 72% of capacity, meaning we need $\sim 1.4\times$ more bandwidth or ~ 2 dB more SNR than the Shannon limit would suggest. This motivates the search for codes that are close to the meta-converse at these short lengths: polar codes with CRC-aided SCL decoding are currently the state of the art for $n \leq 1000$.

Supplementary Derivations

S.1 Proof of the Huffman Optimality via Exchange Argument

We give a complete proof that the Huffman code minimizes expected length $\bar{L} = \sum_i p_i \ell_i$ among all prefix-free codes.

Lemma 1. In any optimal prefix-free code, if $p_i > p_j$ then $\ell_i \leq \ell_j$. *Proof.* Suppose $p_i > p_j$ but $\ell_i > \ell_j$. Swapping codewords: $\Delta\bar{L} = p_i\ell_j + p_j\ell_i - p_i\ell_i - p_j\ell_j = (p_i - p_j)(\ell_j - \ell_i) < 0$, reducing $\bar{L}$. Contradiction. ■

Lemma 2. In any optimal code, the two symbols with smallest probability have codewords of equal maximal length differing only in the last bit. *Proof.* Let x_s, x_t be the two least probable symbols. By Lemma 1, they have the longest codewords. A leaf at depth > 1 must have a sibling (otherwise its parent would be a leaf with shorter codeword, still valid and shorter). So the deepest leaves come in pairs sharing a parent. If x_s and x_t are not siblings, we can reassign x_t's codeword to be the sibling of x_s without increasing $\bar{L}$ (since ℓ_t is unchanged and the displaced symbol has $\leq p_t$ probability). ■

Induction step. Suppose the Huffman algorithm produces an optimal code for any $(M-1)$-symbol source. Given M-symbol source with probabilities $p_1 \geq \cdots \geq p_M$:

1. By Lemma 2, an optimal code has x_{M-1} and x_M as siblings with the longest codewords. Merge them into a single symbol x' with probability $p' = p_{M-1} + p_M$.
2. The reduced $(M-1)$-symbol problem has an optimal code by induction: the Huffman code for the reduced problem.
3. The optimal code for the original problem is obtained by splitting x' into x_{M-1} (codeword $c' \cdot 0$) and x_M (codeword $c' \cdot 1$).
4. The expected length satisfies: $\bar{L}_M = \bar{L}_{M-1} + p_{M-1} + p_M$ (the split adds 1 bit for each of the two symbols). This is exactly what the Huffman algorithm computes. ■

S.2 The Bhattacharyya Parameter and the Z-Domain Recursions

We derive the polarization recursions for a general B-MC using the $\ell^{1/2}$ characterization of the Bhattacharyya parameter.

Define $\mathbf{w} : \mathcal{Y} \to \mathbb{R}^2_{\geq 0}$ by $\mathbf{w}(y) = (W(y|0), W(y|1))$ (the pair of transition probabilities). The Bhattacharyya parameter is then: $Z(W) = \sum_y \sqrt{W(y|0)W(y|1)} = \|\mathbf{w}(\cdot)\|^{1/2}_{\ell^{1/2}}$ (in the $\ell^{1/2}$ quasi-norm sense).

For the W^+ channel (output space $\mathcal{Y}^2 \times \{0,1\}$):

$$
\begin{aligned}
Z(W^+) &= \sum_{y_1,y_2,u_1} \sqrt{W^+(y_1,y_2,u_1|0)W^+(y_1,y_2,u_1|1)} \\
&= \frac{1}{2} \sum_{y_1,y_2} \sum_{u_1=0}^{1} \sqrt{W(y_1|u_1)W(y_1|u_1\oplus 1) \cdot W(y_2|0)W(y_2|1)} \\
&= \frac{1}{2} \left(\sum_{y_2} \sqrt{W(y_2|0)W(y_2|1)} \right) \left(\sum_{y_1} \sum_{u_1} \sqrt{W(y_1|u_1)W(y_1|u_1\oplus 1)} \right) \\
&= \frac{1}{2} Z(W) \cdot 2Z(W) = Z(W)^2.
\end{aligned}
$$

For W^-: $Z(W^-) = \sum_{y_1,y_2} \sqrt{W^-(y_1,y_2|0)W^-(y_1,y_2|1)}$.

By the Cauchy-Schwarz inequality for $\ell^{1/2}$ norms:

$$\begin{aligned} Z(W^-) &= \sum_{y_1,y_2} \sqrt{\left(\sum_{u_2} \frac{1}{2} W(y_1|u_2)W(y_2|u_2)\right)\left(\sum_{u_2} \frac{1}{2} W(y_1|u_2\oplus 1)W(y_2|u_2)\right)} \\ &\le \sum_{y_1,y_2} \frac{1}{2} \sum_{u_2} \sqrt{W(y_1|u_2)W(y_1|u_2\oplus 1)} \cdot W(y_2|u_2) \\ &= \frac{1}{2} \sum_{u_2} \left(\sum_{y_1} \sqrt{W(y_1|u_2)W(y_1|u_2\oplus 1)}\right)\left(\sum_{y_2} W(y_2|u_2)\right) = \frac{1}{2} \cdot Z(W) \cdot 2 = Z(W) \cdot 1, \end{aligned}$$

which gives $Z(W^-) \le 2Z(W) - Z(W)^2$... the exact result follows by a more careful computation. For the BEC, the inequality is tight: $Z(W^-) = 2Z(W) - Z(W)^2$.

S.3 Converse to the Source Coding Theorem (General)

Theorem. For any $(2^{nR}, n)$ lossless source code for i.i.d. source $X \sim p$, if the probability of error $P_e \to 0$, then $R \ge H(X)$.

Proof. Let $g : \mathcal{X}^n \to \{1, \dots, 2^{nR}\}$ be the encoder and $\hat{g}$ the decoder. Define $\tilde{X}^n = \hat{g}(g(X^n))$ the reconstructed sequence. By Fano's inequality:

$$H(X^n|g(X^n)) \le H(X^n|\hat{g}(g(X^n))) + nP_e \log|\mathcal{X}| + h(P_e) = H(X^n|\tilde{X}^n) + nP_e \log|\mathcal{X}| + h(P_e).$$

But $H(X^n|\tilde{X}^n) \le nP_e \log|\mathcal{X}| + h(P_e)$ (Fano). So

$$H(X^n) = H(X^n|g(X^n)) + I(X^n; g(X^n)) \le H(X^n|g(X^n)) + nR,$$

giving $nH(X) = H(X^n) \le 2nP_e \log|\mathcal{X}| + 2h(P_e) + nR$. As $P_e \to 0$: $H(X) \le R$. ■

S.4 Proof of the Blahut-Arimoto Algorithm's Convergence for Channel Capacity

We prove convergence of the Blahut-Arimoto algorithm for channel capacity (dual to the rate-distortion version). The key is identifying it as an instance of *expectation maximization* or *alternating maximization* on a jointly concave objective.

The channel capacity is $C = \max_{p(x)} I(X;Y)$. Define the functional

$$F(p,q) = \sum_{x,y} p(x)W(y|x) \log \frac{W(y|x)}{q(y)},$$

where $q(y)$ is an auxiliary output distribution. Note:

$$\max_q F(p,q) = I(X;Y)|_{p(x)} \quad \text{(achieved by } q(y) = \sum_x p(x)W(y|x)).$$

So $C = \max_{p,q} F(p,q)$. The BA algorithm alternates:

1. $q^{(t+1)}(y) = \sum_x p^{(t)}(x)W(y|x)$ (maximizes F over q for fixed p).
2. $p^{(t+1)}(x) \propto \exp\left(\sum_y W(y|x) \log \frac{W(y|x)}{q^{(t+1)}(y)}\right)$ (maximizes F over p for fixed q, by Gibbs' inequality and Lagrange condition).

Each step cannot decrease F; F is bounded above by $\log|\mathcal{X}|$; therefore F converges. Since F is jointly concave in (p,q) (it is an affine function of q and a concave function of p), the alternating maximization converges to the global maximum C. ■

S.5 The Lovász Local Lemma and Probabilistic Code Constructions

The *Lovász Local Lemma* (LLL) is a powerful probabilistic tool that guarantees the existence of objects with multiple "bad events" avoided simultaneously, provided dependencies between events are limited. It has applications in constructing codes with specific properties.

Lovász Local Lemma (Symmetric Version)

Let $A_1, \dots, A_m$ be events in a probability space. If each event has probability $P[A_i] \leq p$ and is independent of all but d others, then:

$$ep(d+1) \leq 1 \implies P\left[\bigcap_{i=1}^{m} \bar{A}_i\right] > 0.$$

Application: constructing codes avoiding all bad weight-enumerator conditions. Consider a random linear $[n, k]$ code over $\mathbb{F}_2$ (i.e., a random k-dimensional subspace of $\mathbb{F}_2^n$). For each pair of positions $S \subseteq \{1, \dots, n\}$ with $|S| = d$, let A_S be the event that all d columns of $\mathbf{H}$ indexed by S are linearly dependent (i.e., the code has minimum distance $< d+1$). $P[A_S] \leq 2^{-(n-k)(d-k)}$ (roughly). The LLL condition requires that the number of dependencies d_S between events A_S and $A_{S'}$ for $|S \cap S'| \geq 1$ is bounded. When satisfied, the LLL guarantees a code with minimum distance $\geq d+1$, recovering (in a non-constructive way) results near the GV bound.

Numerical Tour: Part 5 — Polar Code Construction for BEC

We trace the complete construction of a $(16, 8)$ polar code for BEC($\epsilon = 0.5$).

Step 1: Recursive Bhattacharyya computation. Starting from $Z_0 = 0.5$ (one channel), recursively apply $Z^- = 2Z - Z^2$ and $Z^+ = Z^2$ for $\ell = 4$ levels:

Index i	Binary path (L/R = $-/+$)	Z_i	Status
1	$----$	$(2(2(2(2 \cdot 0.5 - 0.5^2) - \ldots$	Frozen (high Z)
		... (full tree of 16 values) ...	
16	$++++$	$0.5^{16} = 1.5 \times 10^{-5}$	Information (low Z)

After 4 levels from $Z = 0.5$, the 16 Z-values (in some ordering) are: $\{Z^{----}, Z^{---+}, Z^{--+-}, Z^{--++}, Z^{-+--}, \ldots,$ The values span from $Z^{++++} = (0.5)^{16} \approx 1.5 \times 10^{-5}$ (excellent) to $Z^{----} = 1-(1-0.5)^{16} \approx 0.9999$ (useless).

Step 2: Select information set. Rank 16 channels by Z_i (ascending). The 8 channels with smallest Z_i form $\mathcal{A}$ (information set); the rest are frozen. The information set corresponds to the $+$-heavy paths in the recursion tree.

Step 3: Encoding. $G_{16} = B_{16}(F \otimes F \otimes F \otimes F)$. For information word $\mathbf{u}_{\mathcal{A}} \in \{0,1\}^8$ and frozen bits $\mathbf{u}_{\mathcal{A}^c} = \mathbf{0}$, compute $\mathbf{x} = \mathbf{u}G_{16}$ using the 4-level butterfly structure (16 XOR operations per level, 4 levels = 64 XOR operations total).

Step 4: SC decoding. Received vector $\mathbf{y}$ (over BEC): known bits pass through; erased bits propagate uncertainty. The LLR tree is computed bottom-up: at the leaves, $L_i = +\infty$ (known 0), $-\infty$ (known 1), or 0 (erased). Internal nodes combine using $\phi(L_a, L_b)$ for $-$ nodes and $L_b + (1 - 2\hat{u})L_a$ for $+$ nodes. For BEC, $\phi(L_a, L_b)$: if either is 0 (erased), output 0 (erased); otherwise output L_a if $\text{sign}(L_a) = \text{sign}(L_b)$, else 0 (erased). This is equivalent to the BEC peeling process.

Numerical Tour: Part 6 — Finite Blocklength Bounds for AWGN

Computing the normal approximation at AWGN SNR = 0 dB. $P/N = 1$ (SNR = 0 dB), $C = \frac{1}{2}\log_2 2 = 0.5$ bits/channel use. Dispersion: $V = \frac{1}{2}(1 - 1/(1+1)^2) \cdot (\log_2 e)^2 = \frac{1}{2}(3/4)(2.097) \approx 0.785$ bits^2/channel use (using $\log_2 e \approx 1.443$ bits/nat).

For $n = 200$, $\epsilon = 10^{-3}$ ($Q^{-1}(10^{-3}) \approx 3.09$):

$$R^*(200, 10^{-3}) \approx 0.5 - \sqrt{\frac{0.785}{200}} \cdot 3.09 \approx 0.5 - 0.0627 \cdot 3.09 \approx 0.306 \text{ bits/cu.}$$

For code rate $R = k/n = 0.5$ (same as capacity), actual error probability at $n = 200$:

$$\epsilon \approx Q\left(\frac{(C-R)\sqrt{n}}{\sqrt{V}}\right) = Q(0) = 0.5.$$

This confirms that operating at capacity with $n = 200$ gives 50% error probability — far too high.

To achieve $R = 0.5$ bits/cu with $\epsilon = 10^{-3}$, we need (solving for n):

$$n \geq \left(\frac{\sqrt{V} \cdot Q^{-1}(\epsilon)}{C - R}\right)^2 = \left(\frac{\sqrt{V} \cdot Q^{-1}(\epsilon)}{0}\right)^2 \to \infty.$$

Of course: to achieve the *capacity* itself, one needs $n \to \infty$. For $R = 0.45$ bits/cu (10% back-off) and $\epsilon = 10^{-3}$:

$$n \geq \left(\frac{\sqrt{0.785} \cdot 3.09}{0.5 - 0.45} \right)^2 = \left(\frac{0.886 \cdot 3.09}{0.05} \right)^2 = (54.8)^2 \approx 3000.$$

So at 10% back-off from capacity, one needs $n \approx 3000$ to achieve $\epsilon = 10^{-3}$. This quantifies how blocklength requirements grow as the target error probability decreases or the operating rate approaches capacity.

Comparison: Shannon limit vs. actual modern code performance. At $n = 1000$, rate $R = 0.5$, AWGN SNR $= 0$ dB, $\epsilon_{\text{target}} = 10^{-3}$:

- PPV normal approximation: achievable at $E_b/N_0 \approx 0.5$ dB (back-off ≈ 0.2 dB from Shannon limit).
- 5G NR LDPC (base graph 2), BP 50 iterations: ≈ 0.8 dB (slightly above PPV).
- Polar code (CRC-11, SCL $L = 8$): ≈ 0.7 dB.
- Shannon limit for $n \to \infty$: $E_b/N_0 = -1.6$ dB (binary input, infinite blocks).

Numerical Tour: Part 7 — Slepian-Wolf and Distributed Compression

Practical Slepian-Wolf via channel codes. The connection between Slepian-Wolf coding and channel coding (Wyner's insight, 1974): compressing X at rate $R_X = H(X|Y)$ given side information Y is equivalent to using a channel code for the virtual channel $Y = X \oplus Z$ (where Z is the "error" between X and Y) at rate $1 - H(X|Y) = I(X;Y)$.

Concretely: model X as the "transmitted word" over a virtual BSC with crossover probability $\alpha = P[X \neq Y]$. The "syndrome" of X with respect to a linear code (the product $\mathbf{Hx}$) is transmitted at rate $H(X|Y) = h_b(\alpha)$ bits per source symbol. The decoder uses Y and the syndrome to recover X via channel decoding of the virtual BSC.

Example with $\alpha = 0.1$. $H(X|Y) = h_b(0.1) = 0.469$ bits. Virtual BSC capacity $= 1 - 0.469 = 0.531$ bits. Use a rate 0.531 LDPC code on the virtual BSC. The syndrome has rate $1 - 0.531 = 0.469$ bits per source symbol, which is exactly $H(X|Y)$. Distributed compression at the information-theoretic limit!

Numerical Tour: Part 8 — Reed-Muller Codes

Reed-Muller (RM) codes are a classical family bridging algebraic and polar code theory; they are defined by evaluating multivariate Boolean polynomials and have connections to polar codes via their shared recursive structure.

Reed-Muller Code $\mathrm{RM}(r, m)$

The first-order Reed-Muller code $\mathrm{RM}(1, m)$ has parameters $[2^m, m+1, 2^{m-1}]$: length 2^m, dimension $m+1$, minimum distance 2^{m-1}.
The r-th order Reed-Muller code $\mathrm{RM}(r, m)$ consists of all codewords formed by evaluating multivariate polynomials of degree $\leq r$ in m binary variables at all 2^m points of $\{0,1\}^m$:

$$\mathrm{RM}(r, m) = \left\{ (f(v_1, \ldots, v_{2^m}))_{v \in \{0,1\}^m} : f \in \mathbb{F}_2[x_1, \ldots, x_m],\ \deg f \leq r \right\}.$$

Parameters: $[2^m, \sum_{i=0}^{r} \binom{m}{i}, 2^{m-r}]$.

Recursive structure. $\mathrm{RM}(r,m) = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in \mathrm{RM}(r, m-1),\ \mathbf{v} \in \mathrm{RM}(r-1, m-1)\}$ (the $(\mathbf{u}|\mathbf{u}+\mathbf{v})$ construction). This is exactly the polar code recursion! Indeed, polar codes generalize RM codes by selecting different information sets (by reliability rather than by Hamming weight).

Connection to polar codes. Polar codes for the BEC with $n = 2^m$ have the same generator matrix $G_n = B_n F^{\otimes m}$ as $\mathrm{RM}(r,m)$ (with appropriate B_n), but select information indices by Bhattacharyya parameter rather than Hamming weight. For large m, the optimal polar set for the BEC approaches the $\mathrm{RM}(r,m)$ set, making RM codes near-optimal for the BEC at moderate block lengths.

Example: $\mathrm{RM}(1,3) = [8,4,4]$ **first-order code.** Length $8 = 2^3$, dimension $4 = 3{+}1$, minimum distance $4 = 2^{3-1}$. Generator matrix:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Rows represent the all-ones polynomial ($f = 1$) and the three linear polynomials x_1, x_2, x_3. Minimum distance 4: any non-constant linear polynomial evaluates to 1 on exactly 4 of the 8 inputs. Equivalently, $[8,4,4]$ is a doubly-extended Reed-Solomon code and is related to the $[8,4,4]$ extended Hamming code.

Theoretical Complements: The Geometry of Codes

T.1 The Sphere-Packing Problem and Kissing Numbers

The sphere-packing bound for binary codes has a beautiful geometric interpretation: it asks how many non-overlapping Hamming balls of radius t can be packed into the Hamming cube $\{0,1\}^n$. This is the discrete analogue of the classical sphere-packing problem in Euclidean geometry.

The *kissing number* $\tau(n)$ in Euclidean geometry asks how many unit spheres can simultaneously touch a central unit sphere in $\mathbb{R}^n$. In coding theory, the analogous quantity is the maximum number of codewords of weight $d_{\min}$ in a perfect code. For the Hamming code, every ball of radius 1 is "kissed" by exactly n other balls (one per neighbor), matching the Hamming ball volume $V(n,1) = 1+n$.

The connection deepens for Leech lattice codes: the $[24, 12, 8]$ Golay code (the binary perfect code beyond Hamming codes) is related to the Leech lattice in $\mathbb{R}^{24}$, which achieves the densest known sphere packing in 24 dimensions with kissing number 196560. This is an example of how coding theory and geometry of numbers are deeply intertwined.

T.2 Algebraic Geometry Codes

Beyond Reed-Solomon codes, the most powerful algebraic codes are *algebraic geometry* (*AG*) *codes* (Goppa, 1981), constructed from algebraic curves over finite fields.

AG Code (Brief Statement)

Let X be an algebraic curve of genus g over $\mathbb{F}_q$, with rational points $P_1, \dots, P_n \in X(\mathbb{F}_q)$ and a divisor G (formal sum of points) of degree $\deg G = d < n$. The AG code is:

$$C(X, P, G) = \{(f(P_1), \dots, f(P_n)) : f \in \mathcal{L}(G)\} \subseteq \mathbb{F}_q^n,$$

where $\mathcal{L}(G)$ is the Riemann-Roch space of functions with poles bounded by G. The code has parameters $[n, k, d^*]$ with $k \geq \deg G - g + 1$ and $d^* \geq n - \deg G$.

The *Tsfasman-Vladut-Zink* (*TVZ*) *bound* (1982) shows that AG codes over $\mathbb{F}_{q^2}$ for $q = p^2 \geq 49$ achieve rates above the GV bound:

$$R \geq 1 - \delta - \frac{1}{\sqrt{q} - 1},$$

beating the GV bound $1 - h_b(\delta)$ for large enough q and specific δ. This was the first demonstration that codes can beat the GV bound over large alphabets, resolving a long-standing question.

T.3 List Decoding and the Guruswami-Sudan Algorithm

List decoding asks for all codewords within a given radius from the received word (not just the single closest). This allows decoding beyond the unique-decoding radius $t = \lfloor (d-1)/2 \rfloor$, at the cost of producing a list of candidate codewords.

Johnson bound for list decoding. For any code with minimum distance d, list decoding up to radius $e < n(1 - \sqrt{1 - d/n})$ (the Johnson radius) produces a list of size polynomial in n.

Guruswami-Sudan (GS) algorithm. For Reed-Solomon codes, the GS algorithm (1999) achieves list decoding up to the Johnson radius $e_{\text{JS}} = n - \sqrt{nk}$ (unique decoding only handles $e < (n-k)/2$). The algorithm runs in polynomial time and consists of:

1. *Interpolation step*: find a low-degree bivariate polynomial $Q(x, z)$ vanishing with high multiplicity at each received point (α_i, y_i).
2. *Root-finding step*: find all polynomials $f(x)$ of degree $< k$ such that $Q(x, f(x)) \equiv 0$, using factorization over $\mathbb{F}_q[x]$.

The number of solutions is at most $O(\sqrt{n/k})$ (polynomial in n). GS list decoding achieves rates approaching capacity for worst-case channels, and is used in Reed-Solomon decoding implementations for storage and deep-space communications.

T.4 Expander Codes

Expander codes (Sipser and Spielman, 1996) use the expansion properties of bipartite graphs to construct linear codes with linear minimum distance and $O(n)$ encoding/decoding time.

Expander Code

Let $G = (V \cup C, E)$ be a bipartite (Δ, γ)-expander: every $S \subseteq V$ with $|S| \leq \delta n$ has neighborhood $|N(S)| \geq (1 - \epsilon)\Delta|S|$ for small ϵ. The *expander code* with respect to G and a local code $\mathcal{C}_0$ (small inner code) has codewords $\mathbf{x}$ such that for each check node $c \in C$, the restriction of $\mathbf{x}$ to $N(c)$ is a codeword of $\mathcal{C}_0$.

If $\mathcal{C}_0$ is a $[\Delta, k_0, d_0]$ inner code with $d_0 > \Delta/2$, then the expander code has minimum distance $\Omega(n)$ (linear). Encoding is $O(n)$ (local inner code encoding at each check node). Decoding uses an iterative "flipping" algorithm: repeatedly flip any bit that is in more unsatisfied checks than satisfied ones. The expansion property ensures that errors are corrected in $O(\log n)$ rounds.

I.1 The Data-Processing Inequality in Coding

The *data-processing inequality* (DPI) states that for any Markov chain $X \to Y \to Z$: $I(X;Z) \leq I(X;Y)$. In coding theory, the DPI appears in converse proofs:

If encoder $f : \mathcal{X}^n \to \{1, \dots, M\}$ and decoder $g : \{1, \dots, M\} \to \hat{\mathcal{X}}^n$ form a Markov chain $X^n \to f(X^n) \to \hat{X}^n$, then:

$$I(X^n; \hat{X}^n) \leq I(X^n; f(X^n)) \leq H(f(X^n)) \leq \log M.$$

This gives the converse to the source-channel separation theorem: the rate $R = \log M/n$ must satisfy $R \geq H(X)$ for lossless coding.

I.2 Entropy Power Inequality

The *entropy power inequality* (EPI, Shannon 1948) is the most important information inequality for Gaussian channels:

$$e^{2h(X+Y)} \geq e^{2h(X)} + e^{2h(Y)},$$

for independent continuous random variables X and Y, where h is differential entropy in nats. Equality holds iff X and Y are Gaussian.

Application to AWGN converse. The EPI is the key ingredient in the converse to the AWGN capacity theorem. Suppose $Y = X + Z$ with $Z \sim \mathcal{N}(0, N)$ independent of X. For any X with $\mathbb{E}[X^2] \leq P$:

$$e^{2h(Y)} = e^{2h(X+Z)} \geq e^{2h(X)} + e^{2h(Z)} \geq 2\pi eP + 2\pi eN = 2\pi e(P+N).$$

But Gaussian maximizes entropy: $h(Y) \leq \frac{1}{2}\log(2\pi e(P+N))$, with equality when $X \sim \mathcal{N}(0, P)$. The EPI shows this upper bound is tight and is achieved exactly by Gaussian inputs, confirming the capacity formula.

I.3 Tensorization and the Augmented Channel

Tensorization of entropy: for product distributions $P_{X^n} = \prod P_{X_i}$, $H(X^n) = \sum H(X_i)$. This is used ubiquitously in information theory to reduce n-letter coding problems to single-letter optimizations.

The analogous property fails for mutual information in general channels: $I(X^n; Y^n) \neq \sum_i I(X_i; Y_i)$ for correlated inputs. The correct tensorization is: $I(X^n; Y^n) = \sum_i I(X_i; Y_i | Y_1^{i-1})$ (chain rule), which depends on the conditional distribution of Y_i given previous outputs.

For memoryless channels: $I(X^n; Y^n) = \sum_i I(X_i; Y_i)$ iff the inputs are independent, which is the case for random Gaussian codebooks. The tensorization property is what allows the AWGN capacity proof to reduce to a single-letter formula.

Problems and Exercises

The following problems develop the material of each Part in greater depth. They are organized by difficulty: ($\star$) straightforward; ($\star\star$) requires combining several ideas; ($\star\star\star$) challenging, closer to research level.

Part 1: Block Codes

1.1 ($\star$) Show that the $[7, 4, 3]$ Hamming code is equivalent (up to column permutation) to the cyclic code generated by $g(x) = 1 + x + x^3$ over $\mathbb{F}_2$.

1.2 ($\star\star$) For $n = 2^r - 1$, prove that the $r \times n$ Hamming parity-check matrix $\mathbf{H}$ (with columns = binary representations of $1, \dots, 2^r - 1$) has full rank r over $\mathbb{F}_2$. [Hint: show the r standard basis vectors appear as columns.]

1.3 ($\star\star$) Compute $A(8, 4)$, the maximum number of binary codewords with block length 8 and minimum distance 4. Show that the Hamming and Plotkin bounds give $A(8, 4) \le 20$, and construct a code achieving $A(8, 4) = 20$ (the shortened Reed-Muller code).

1.4 ($\star$) Verify the MacWilliams identity for the $[4, 2, 2]$ even-weight code: $\mathcal{C} = \{0000, 1100, 0011, 1111, 1010, 010$ $\{$even weight$\}$. Compute $W_{\mathcal{C}}(x, y)$ and $W_{\mathcal{C}^\perp}(x, y)$ directly, and verify the identity.

1.5 ($\star\star\star$) The *Delsarte LP bound* for $A(5, 3)$: formulate the LP in terms of Krawtchouk polynomials $K_0, K_1, K_2, K_3, K_4, K_5$ for $n = 5$, and solve it to show $A(5, 3) \le 4$. Verify this is tight by exhibiting a $(5, 4, 3)$ code.

Part 2: Linear and Cyclic Codes

2.1 ($\star$) Find the generator polynomial $g(x)$ and parity-check polynomial $h(x)$ for the $[7, 4, 3]$ Hamming code, viewed as a cyclic code over $\mathbb{F}_2$. [Answer: $g(x) = 1 + x + x^3$, $h(x) = 1 + x + x^2 + x^4$.]

2.2 ($\star\star$) Construct the RS$(6, 4)$ code over $\mathbb{F}_7$ with evaluation points $\{1, 2, 3, 4, 5, 6\}$. Compute the codeword for information polynomial $f(x) = 3 + 2x + x^2 + 4x^3$, verify the minimum distance is 3, and exhibit two codewords at distance 3.

2.3 ($\star\star$) Prove that the dual of a q-ary MDS $[n, k, n - k + 1]$ code is again an MDS code $[n, n - k, k + 1]$. [Hint: use the MacWilliams identity and the weight distribution of an MDS code.]

2.4 ($\star$) Implement (on paper) the Berlekamp-Massey algorithm for the syndrome sequence $S = (1, 0, 1, 1)$ over $\mathbb{F}_2$. Find the shortest LFSR generating this sequence.

2.5 ($\star\star\star$) Prove the BCH bound for a cyclic code with generator polynomial having consecutive roots $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+\delta-2}$. State precisely when the bound is tight (i.e., when $d_{\min} = \delta$), and give an example where the true $d_{\min} > \delta$.

Part 3: LDPC and Turbo Codes

3.1 ($\star$) For a $(3, 6)$-regular LDPC code, compute the BP threshold on the BEC by finding the non-trivial fixed point of $p = \epsilon\lambda(1 - \rho(1 - p))$ with $\lambda(x) = x^2$, $\rho(x) = x^5$.

3.2 ($\star\star$) Derive the check-to-variable message update rule in the LLR domain from the probability-domain sum-product rule. Show the equivalence of the tanh and f-function forms.

3.3 ($\star\star$) For turbo codes with BCJR decoding, show that the extrinsic LLR $L_e = L_{\text{post}} - L_a -$

L^{ch} correctly separates information from the component code and avoids double-counting when passed to the other decoder.

3.4 ($\star\star\star$) Design an irregular LDPC code for the BEC with rate $1/2$ and threshold $\epsilon^* \geq 0.48$. Use degree distributions λ and ρ that satisfy the density evolution threshold condition. [Hint: start from the known optimization of Luby et al. (2001).]

Part 4: Data Compression

4.1 ($\star$) Encode the string "MISSISSIPPI" using LZ78 (with an initially empty dictionary except for single characters). List all dictionary entries added during encoding and the corresponding output codes.

4.2 ($\star\star$) For a Gaussian source $X \sim \mathcal{N}(0, 4)$ with MSE distortion, compute $R(D)$ for $D \in \{0.1, 0.5, 1.0, 2.0, 4.0\}$. Plot the rate-distortion curve and identify the point where $R(D) = 0$.

4.3 ($\star\star$) Run 5 iterations of the Blahut-Arimoto rate-distortion algorithm for a binary source $X \sim \text{Bern}(0.3)$ with Hamming distortion $d(x, \hat{x}) = \mathbf{1}\{x \neq \hat{x}\}$ and slope $s = -2$. What is the achieved (D, R) pair? Compare to the theoretical rate-distortion function $R(D) = h_b(0.3) - h_b(D)$ for $D \leq 0.3$.

4.4 ($\star\star\star$) Prove the converse to the rate-distortion theorem for a binary source with Hamming distortion. Show that any $(2^{nR}, n)$ code achieving expected distortion D must have $R \geq R(D) = h_b(p) - h_b(D)$ for $D \leq p \leq 1/2$.

Part 5: Kolmogorov Complexity

5.1 ($\star$) Show that $K(x^n) = K(x) + O(\log n)$ for $x^n = xx\cdots x$ (string x repeated n times). What does this say about the compressibility of periodic sequences?

5.2 ($\star\star$) Use the incompressibility method to prove that the number of steps in the Euclidean algorithm on inputs (a, b) with $a > b > 0$ is $O(\log \min(a, b))$. [Hint: apply incompressibility to a random pair (a, b).]

5.3 ($\star\star\star$) Prove the chain rule for Kolmogorov complexity: $K(x, y) = K(x) + K(y|x, K(x)) + O(\log K(x, y))$. [Note: the conditioning on $K(x)$ is necessary for exact equality; explain why.]

5.4 ($\star\star$) The Mandelbrot set $M \subset \mathbb{C}$ has Kolmogorov complexity $K(M) = O(1)$ (it is computable from a short program). Yet its boundary has Hausdorff dimension 2 (it is maximally complex geometrically). Explain this apparent paradox using the distinction between $K(M)$ (complexity of the set as a mathematical object) and $K(B(M, \epsilon))$ (complexity of a finite approximation at resolution ϵ).

Part 10 Lattice Codes and Polar Lattices

Sections 10.1 – 10.4. Lattices in $\mathbb{R}^n$: definitions, fundamental regions, packing density; the quantisation theorem and Zador's asymptotics; Loeliger's Construction A; the AWGN capacity via lattice codes; Erez–Zamir nested lattice codes; polar lattices achieving the AWGN capacity.

10.1 Lattices: Definitions and Basic Properties

Lattice Fundamentals

A *lattice* $\Lambda \subset \mathbb{R}^n$ is a discrete additive subgroup: $\Lambda = \{\mathbf{Bz} : \mathbf{z} \in \mathbb{Z}^n\}$ for some non-singular *generator matrix* $\mathbf{B} \in \mathbb{R}^{n\times n}$. The *fundamental parallelotope* (or Voronoi cell of the origin) is:

$$\mathcal{V}(\Lambda) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \le \|\mathbf{x} - \boldsymbol{\lambda}\| \text{ for all } \boldsymbol{\lambda} \in \Lambda\}.$$

The *fundamental volume* is $V(\Lambda) = |\det(\mathbf{B})|$, equal to the volume of any fundamental domain.

Three key parameters of a lattice Λ:

- *Packing radius* $r_{\text{pack}}(\Lambda) = \frac{1}{2}d_{\min}(\Lambda)$, where $d_{\min}(\Lambda) = \min_{\mathbf{0}\neq\boldsymbol{\lambda}\in\Lambda} \|\boldsymbol{\lambda}\|$ is the minimum distance.
- *Covering radius* $r_{\text{cov}}(\Lambda) = \max_{\mathbf{x}\in\mathbb{R}^n} \min_{\boldsymbol{\lambda}\in\Lambda} \|\mathbf{x} - \boldsymbol{\lambda}\|$: the smallest r such that the spheres $B(\boldsymbol{\lambda}, r)$ cover all of $\mathbb{R}^n$.
- *Packing density* $\delta(\Lambda) = \frac{\text{Vol}(B(0,r_{\text{pack}}))}{V(\Lambda)} = \frac{r_{\text{pack}}^n \pi^{n/2}/\Gamma(n/2+1)}{V(\Lambda)}$.

The *sphere-packing problem* asks for the lattice maximising $\delta(\Lambda)$ in $\mathbb{R}^n$. Solved in $n = 1, 2, 3, 8, 24$: the Leech lattice Λ_{24} achieves $\delta \approx 0.0019$ in $\mathbb{R}^{24}$, which is astronomically dense compared to random packings. In high dimensions, the best known lattices (including Barnes–Wall Λ_{16} and Leech Λ_{24}) achieve exponentially better packing than random configurations.

Construction A: Codes to Lattices

The *Construction A* (Leech and Sloane, 1971) lifts a binary linear code to a lattice:

Construction A

Let $\mathcal{C}$ be an $[n, k, d]_2$ binary linear code. Define:

$$\Lambda_A(\mathcal{C}) = \frac{1}{\sqrt{2}}\{\mathbf{x} \in \mathbb{Z}^n : \mathbf{x} \bmod 2 \in \mathcal{C}\}.$$

This is an n-dimensional lattice with fundamental volume $V = 2^{(n-k)/2}$, minimum distance $d_{\min} = \min(d/\sqrt{2}, 1/\sqrt{2}\cdot 2) = \min(\sqrt{d/2}, \sqrt{2})$, and 2^k cosets of $2\mathbb{Z}^n$ in $\Lambda_A(\mathcal{C})$ corresponding to the codewords.

Construction A establishes a bridge between algebraic coding theory and lattice geometry. The Gosset lattice E_8 arises from the $[8, 4, 4]_2$ extended Hamming code; the Leech lattice Λ_{24} arises from the $[24, 12, 8]_2$ extended Golay code. Both are among the densest known packings in their dimensions.

Lattice Quantisation and the MMSE Distortion

A lattice Λ defines a quantiser: map $\mathbf{x} \in \mathbb{R}^n$ to the nearest lattice point $Q_\Lambda(\mathbf{x}) = \arg\min_{\boldsymbol{\lambda}\in\Lambda} \|\mathbf{x} - \boldsymbol{\lambda}\|$. The quantisation error is $\mathbf{e} = \mathbf{x} - Q_\Lambda(\mathbf{x}) \in \mathcal{V}(\Lambda)$. For a uniform source on $\mathbb{R}^n$ (approximated by a Gaussian), the mean squared error per dimension is:

$$D = \frac{1}{n}\int_{\mathcal{V}} \|\mathbf{x}\|^2 \, d\mathbf{x} \,/\, V(\Lambda) = G(\Lambda) \cdot V(\Lambda)^{2/n},$$

where $G(\Lambda)$ is the *normalised second moment* (NSM) of the fundamental domain. For large n, the optimal lattice has NSM approaching the NSM of an n-sphere: $G_{\text{sphere}}(n) = n/(2\pi e \cdot (n+2)) \to 1/(2\pi e)$ as $n \to \infty$.

Nested Lattice Codes for the AWGN Channel

Erez and Zamir (2004) showed that *nested lattice codes* (lattice coset codes) achieve the Gaussian channel capacity:

Theorem — Erez–Zamir: Lattice Codes Achieve AWGN Capacity

For the AWGN channel $Y = X + Z$ with power constraint P and $Z \sim \mathcal{N}(0, N)$, there exist nested lattice pairs $\Lambda_c \subset \Lambda_f$ (fine lattice Λ_f inside coarse lattice Λ_c) with $V(\Lambda_c)/V(\Lambda_f) = 2^{nR}$ such that:

1. Λ_f is good for channel coding: the probability that Z exits the Voronoi cell of Λ_f is at most ϵ for large n (approaching the Gaussian error exponent).
2. Λ_c is good for source coding: the NSM of Λ_c approaches $1/(2\pi e)$.
3. The rate $R = \frac{1}{2}\log_2(1 + P/N) = C$ is achievable with vanishing error.

The dithered lattice code. A random dither $\mathbf{d}$ (uniform on $\mathcal{V}(\Lambda_c)$, known to both encoder and decoder) is used:

1. Encoder: given message index m, select coset representative $\mathbf{t}_m \in \Lambda_f/\Lambda_c$, compute $\mathbf{x} = [\mathbf{t}_m - \mathbf{d}] \bmod \Lambda_c$ (ensures uniform distribution and power constraint).
2. Decoder: given $\mathbf{y} = \mathbf{x} + \mathbf{z}$, compute $Q_{\Lambda_f}(\mathbf{y} + \mathbf{d}) \bmod \Lambda_c$. By the crypto lemma ($\mathbf{x} + \mathbf{d}$ is uniform on $\mathcal{V}(\Lambda_c)$), the effective noise is $\mathbf{z} - Q_{\Lambda_c}(\mathbf{z})$, with power approaching that of a Gaussian of variance N as $n \to \infty$.

Polar lattices. A recent construction (Liu and Ling, 2019) builds Λ_f using a polarisation argument on the lattice structure, obtaining explicit lattice codes with $O(n \log n)$ complexity that provably achieve AWGN capacity.

10.2 The Sphere-Packing Bound for Lattices

Minkowski–Hlawka Theorem

The *Minkowski–Hlawka* theorem provides an existential lower bound on lattice packing density, analogous to the Gilbert–Varshamov bound in coding theory:

Theorem — Minkowski–Hlawka (1943)

For every $n \geq 1$, there exists a lattice $\Lambda \subset \mathbb{R}^n$ with packing density

$$\delta(\Lambda) \;\geq\; \frac{\zeta(n)}{2^{n-1}},$$

where $\zeta(n)$ is the Riemann zeta function. For large n: $\delta \geq 2^{-n}(1+o(1))$.

This is a non-constructive bound. Constructive lattices (like Leech) approach $2^{-n/2}$, exponentially better, by exploiting algebraic structure. The best known upper bound (Rogers, Kabatiansky–Levenshtein) gives $\delta \leq 2^{-0.599n(1+o(1))}$; the gap between 2^{-n} (Minkowski) and $2^{-0.599n}$ (Rogers) in the exponent is a major open problem.

The Elias–Bassalygo Comparison

The Hamming sphere-packing bound for binary codes gives an upper bound on code size: $M \leq 2^n / V(n,t)$ where $V(n,t)$ is the volume of a binary Hamming ball of radius t. The lattice packing problem is the continuous analogue: how many non-overlapping unit spheres fit in a unit cube (per unit volume)?

For large n, the packing density of Euclidean spheres is $\delta \leq 2^{-(0.599\ldots)n}$ (Kabatiansky–Levenshtein), while the best constructions achieve $\delta \approx 2^{-n/2}$. Bridging this gap is one of the deepest open problems in mathematics and has implications for the design of high-dimensional quantisers and multiple-antenna communication systems.

10.3 Lattice Coding for Gaussian Multiple-Access Channels

Integer-Forcing Receivers

In a K-user Gaussian MAC with channels $Y = \sum_{k=1}^{K} h_k X_k + Z$, the classical approach decodes each message separately. *Integer-forcing* (Zhan et al., 2014) instead decodes K integer linear combinations $\mathbf{a}_1, \ldots, \mathbf{a}_K \in \mathbb{Z}^K$ of the lattice codewords, then inverts the integer matrix:

Integer-Forcing Receiver

Choose integer matrix $\mathbf{A} \in \mathbb{Z}^{K \times K}$ with $\det(\mathbf{A}) \neq 0$. The receiver computes $\hat{\mathbf{b}}_k = Q_\Lambda(\mathbf{a}_k^\top [\mathbf{y} + \mathbf{d}])$ for each row $\mathbf{a}_k$ of $\mathbf{A}$, recovering K integer linear combinations of lattice codewords. Then $\hat{\mathbf{t}} = \mathbf{A}^{-1}\hat{\mathbf{b}}$ recovers the original messages (modulo Λ_c).

The rate achievable by integer-forcing depends on $\mathbf{a}_k$ and $\mathbf{h} = (h_1, \ldots, h_K)$:

$$R_k \leq \frac{1}{2} \log_2 \left(\frac{P}{\|\mathbf{A}\mathbf{h}^{-1}\|^2_{(\mathbf{h})} + 1/\mathrm{SNR}} \right),$$

where the quadratic form involves the effective channel seen by the k-th integer combination. The optimal $\mathbf{A}$ is found by lattice reduction (LLL algorithm), making integer-forcing computationally efficient even for large K.

Compute-and-Forward Protocol

The *compute-and-forward* protocol (Nazer and Gastpar, 2011) uses lattice codes to enable relay nodes in a network to decode linear combinations of transmitted messages over the air, exploiting the additive structure of the wireless channel. Each relay computes an integer linear combination of the source codewords (without full decoding of individual messages) and forwards it; the destination collects enough such combinations to recover all messages.

Compute-and-forward achieves rates close to the multiple-access capacity for many channel configurations, and is particularly effective when the channel coefficients h_k are approximately rational (since then integer combinations can have small effective noise).

10.4 The Relationship Between Lattice Codes and Polar Codes

Polar lattices are constructed by applying the polar code transform to the component binary codes in a multilevel lattice construction. The key insight is that polarisation extends to continuous-alphabet channels: the split-and-combine operation on continuous Gaussian channels produces synthetic channels that polarise towards either infinite capacity or zero capacity, just as for binary channels.

Formally: let $\Lambda = \Lambda_A(\mathcal{C})$ be a Construction-A lattice from a binary polar code $\mathcal{C}$. The multilevel encoding sends each level of the binary polar encoder to a different quantisation level. The SC decoder operates level by level, using soft information passed between levels, analogously to SC decoding of binary polar codes. The resulting scheme achieves the AWGN capacity with $O(n \log n)$ complexity — the first explicit, efficient AWGN capacity-achieving lattice code.

> The construction of polar lattices unifies the two main threads of coding theory: algebraic structure (from lattices and Construction A) and probabilistic design (from polar codes and density evolution). This synthesis is the lattice analogue of the turbo/LDPC revolution for binary channels — it shows that capacity-approaching performance at continuous-alphabet channels can be achieved with structured, efficiently decodable codes.

Sections 11.1 – 11.4. The flash memory channel: asymmetric errors, charge leakage, and over-programming; rank-modulation and Gray codes for flash; NAND flash ECC in practice; DNA storage: the synthesis/sequencing channel, combinatorial codes for DNA, the Fountain-DNA approach.

11.1 The Flash Memory Channel

NAND Flash Physics

A NAND flash memory cell stores a charge q on a floating gate. In a *single-level cell* (SLC), charge is above or below a threshold θ, encoding a single bit. In a *multi-level cell* (MLC), charge takes one of $q = 2^b$ levels (for b bits per cell). The read threshold at level i is $\theta_i = i \cdot \Delta + \sigma Z_i$ where Δ is the nominal level spacing, σ quantifies threshold voltage variation, and $Z_i \sim \mathcal{N}(0,1)$ is noise.

Error modes in NAND flash:

- *Charge gain (over-programming):* programming overshoot $\to$ level is too high. Probability $\approx Q((\Delta - |\delta|)/\sigma)$ for overshoot δ.
- *Charge loss (retention):* over time (t years), charge leaks: $q(t) \approx q(0) - c\log(1 + t/t_0)$. After 10 years, drift can exceed 0.2Δ.
- *Read disturb and write-erase cycling:* repeated use widens the threshold voltage distribution $\sigma \to \sigma(1 + \alpha N_{\text{cycles}})$.
- *Cell-to-cell interference:* parasitic capacitive coupling shifts levels of adjacent cells when programming.

The dominant error model is an *asymmetric sliding-window channel*: errors are more likely to shift a level upward (by over-programming) than downward (charge loss is slow on the programming timescale). This asymmetry motivates *rank-modulation* codes.

Rank Modulation

Rank modulation (Jiang et al., 2008) stores information in the *relative order* of cell charges rather than their absolute values. A set of n cells stores a permutation $\pi \in S_n$: cell i has the $\pi(i)$-th largest charge. Since only relative order matters, charge leakage (which affects all cells roughly equally) does not change π until the relative order changes.

Rank-Modulation Code

A *rank-modulation code* is a subset $\mathcal{C} \subseteq S_n$ (a set of permutations). The information content is $\log_2 |\mathcal{C}|$ bits, the rate is $R = (\log_2 |\mathcal{C}|)/n$ bits per cell, and the minimum distance is

$$d_{\min}(\mathcal{C}) = \min_{\pi \neq \sigma \in \mathcal{C}} d_{\text{Kendall}}(\pi, \sigma),$$

where $d_{\text{Kendall}}(\pi,\sigma) = |\{(i,j) : i < j, \pi^{-1}(i) < \pi^{-1}(j) \text{ but } \sigma^{-1}(i) > \sigma^{-1}(j)\}|$ is the number of pairwise disagreements (Kendall's τ-distance). Kendall's distance counts the minimum number of adjacent transpositions to go from π to σ.

The analogue of the Singleton bound for permutation codes is: $|\mathcal{C}| \le n!/(k!)$ for $d_{\min} \ge n - k + 1$ (Deza–Frankl bound). The analogue of MDS codes is $|\mathcal{C}| = n!/k!$: codes achieving

this are called *perfect* permutation codes.

Rewriting Codes and WOM

A *write-once memory* (WOM) code (Rivest and Shamir, 1982) allows multiple information-rewrite operations on a memory where cells can only increase their level (as in flash, where erasing the entire block before rewriting is expensive). A t-write WOM code allows t message rewrites using only cell level increases:

Theorem — WOM Capacity (Rivest–Shamir)

For a binary WOM with t writes and n cells, the maximum total number of bits that can be reliably written across all t writes is:

$$n \cdot \sum_{i=1}^{t} \frac{1}{i} \log_2 \binom{n}{i/n} + o(n),$$

or equivalently, the WOM rate is $\sum_{j=1}^{t} h(p_j)$ where p_j are optimally chosen. For $t = 2$ and binary cells: WOM rate $= h(1/3) + h(1/2) \approx 0.918 + 1 = 1.918$ bits per cell, exceeding the "naive" limit of 1 bit per cell.

11.2 ECC for NAND Flash in Practice

Soft-Decision Decoding for Flash

Modern NAND flash ECC stacks use *soft-decision* information: instead of a hard binary decision (0 or 1 per bit), the flash controller reads the cell charge at multiple threshold levels to produce LLRs $L = \log P(\text{bit} = 0|q)/P(\text{bit} = 1|q)$. For a symmetric Gaussian model:

$$L = \frac{2q}{\sigma^2} \quad \text{(for a SLC cell with charges } \pm 1 \text{ and noise variance } \sigma^2\text{)}.$$

Flash ECC evolution:

Era	Code	Overhead
2000–2010	BCH$(1023, 933, t = 11)$ (SLC)	9%
2010–2015	BCH$(4096, 4069, t = 40)$ (MLC/TLC)	7%
2015–2020	LDPC (QC, rate 0.87–0.92)	8–15%
2020–present	LDPC + Polar hybrid, 5G-inspired	10–20%

Threshold voltage distribution model. The cell-level distribution for a 4-level (2-bit, MLC) NAND cell can be modelled as a mixture of Gaussians:

$$p(v) = \frac{1}{4} \sum_{l=0}^{3} \mathcal{N}(v; \mu_l, \sigma^2(N_{\text{cycles}})),$$

where μ_l are the nominal levels and $\sigma^2(N_{\text{cycles}})$ grows with cycling. The optimum hard-decision thresholds θ_l^* are found by minimising the cross-level error probability, and the soft LLRs are computed from $p(v)$ directly.

The Flash Channel as an Information Channel

Viewed information-theoretically, the MLC flash cell defines a multi-level channel with input $X \in \{0, 1, 2, 3\}$ and output $Y = X + Z_{\text{total}}$ where Z_{total} captures charge variation, retention

loss, and cell-to-cell interference. The symmetric capacity:

$$C_{\text{flash}}(N_{\text{cycles}}) = H(Y) - H(Y|X) = \log_2 4 - h(\sigma^2(N_{\text{cycles}})/\Delta^2),$$

decreases monotonically as cycling increases σ. When $C_{\text{flash}} < R_{\text{ECC}}$, the block must be retired (taken offline). The ECC designer balances initial overhead ($\propto 1 - R_{\text{ECC}}$) against the block retirement lifetime.

11.3 DNA Data Storage: Information Theory

The DNA Storage Channel

DNA data storage encodes digital information in synthetic DNA sequences and retrieves it via next-generation sequencing (NGS). A DNA "file" consists of M oligonucleotides (short single-stranded DNA molecules) of length L base pairs (bp) each, drawn from a 4-letter alphabet $\{A, C, G, T\}$, stored in an unordered pool.

Channel characteristics:

- *Substitution errors:* sequencing error rate $\epsilon_s \approx 10^{-3}$–$10^{-2}$ per base (Illumina short-read sequencing), with non-uniform error profile.
- *Insertion/deletion (indel) errors:* more common in long-read sequencers (Oxford Nanopore: $\epsilon_{\text{indel}} \approx 3$–$5\%$ per base), creating synchronisation problems.
- *Coverage randomness:* each oligonucleotide is read C times on average, where $C \sim \text{Poisson}(\lambda)$. Some oligos may not be read at all (missing sequences, probability $e^{-\lambda}$).
- *Strand loss and degradation:* over long storage times (years, centuries), oligos degrade. The fraction surviving after time t is approximately $e^{-t/\tau}$.

Information Capacity of DNA Storage

Assume each of M oligos has length L nt drawn from $\{A, C, G, T\}$. The information content is at most $ML \log_2 4 = 2ML$ bits. The effective capacity, accounting for:

- Ordering loss (unordered pool): subtract $\log_2 M!/\log_2 4^{ML}$ fraction for index bits needed to reorder oligos: $\approx M \log_2 M/(2ML) = \log_2 M/(2L)$. For $L = 200$ and $M = 10^6$: index overhead $\approx 10\%$.
- Coverage randomness: oligos not read are erasures. With λ coverage, fraction $e^{-\lambda}$ are missing; their information is lost. Effective rate loss: $\approx e^{-\lambda}$ (small for $\lambda \geq 5$).
- Biochemical constraints: homopolymer runs > 3 nt reduce synthesis fidelity; GC content must be 40–60%. These constraints reduce capacity by ≈ 10–15%.

The net capacity for practical DNA storage is approximately 1.5–1.8 bits/nt, vs. the theoretical 2 bits/nt.

Coding Schemes for DNA Storage

Goldman et al. (2013): Overlapping windowing scheme. Each byte of data is encoded as a triplet of nt (using a Huffman-like code for DNA), with $4\times$ redundancy and Hamming error correction. Achieved ≈ 1.0 bits/nt effective density.

Grass et al. (2015): RS codes with inner Reed–Solomon code over $\mathbb{F}_4$ (the four DNA bases) for error correction within each oligo, and outer RS code across oligos for erasure correction. Achieved 0.85 bits/nt, with data retrieved intact after 2000 years of estimated degradation at $9.4°$C.

Organick et al. (2018): Fountain-code based approach. Each oligo carries a fountain code packet (random linear combination over $\mathbb{F}_{256}$ of data blocks, encoded into 4-base codons). The decoder uses belief propagation peeling. With $3\times$ coverage, ≈ 214 petabytes/g theoretical density, demonstrated at 200 MB/g. Rate: ≈ 1.5 bits/nt.

Insertion-Deletion-Substitution Codes for DNA

The *Levenshtein distance* $d_L(u, v)$ is the minimum number of insertions, deletions, and substitutions to convert string u to string v. Codes correcting t indels must have $d_L \geq 2t + 1$.

The *Varshamov–Tenengolts* (VT) *codes* (1965) are the fundamental binary codes correcting a single deletion or insertion. For n-bit strings:

$$\mathrm{VT}_a(n) = \{x \in \{0,1\}^n : \sum_{i=1}^{n} i \cdot x_i \equiv a \pmod{n+1}\}.$$

$|\mathrm{VT}_a(n)| \approx 2^n/(n+1)$, so $R = 1 - \log_2(n+1)/n \to 1$ as $n \to \infty$. VT codes are nearly optimal: the known lower bound on single-deletion-correcting code size is $2^n/(n+1)$ (Levenshtein, 1966). The extension to multiple deletions (Helberg codes) and to DNA alphabets ($q = 4$) is an active research area.

11.4 Information-Theoretic Limits of DNA Storage

The DNA Storage Channel Capacity

Model the DNA storage system as a channel from input (synthesised sequences) to output (sequenced sequences), parameterised by coverage λ, substitution rate ϵ_s, indel rate ϵ_i, and constraint set $\mathcal{C}_{\text{bio}}$ (biochemical constraints). The storage capacity is:

$$C_{\text{DNA}} = \max_{P_{\text{seq}} \text{ on } \mathcal{C}_{\text{bio}}} I(\text{sequence pool; sequenced reads}).$$

This is a compound channel (unknown coverage at synthesis time) with erasures (missing oligos) and substitutions (within-oligo errors). By the Slepian–Wolf theorem, the ordering information (which oligo maps to which position) need not be stored in the sequence content if the decoder uses joint decoding of all reads — a significant saving in index overhead.

Capacity bounds. For the substitution-only model with coverage λ:

$$C_{\text{DNA}} \approx (1 - e^{-\lambda})(2 - h_4(\epsilon_s)) \text{ bits/nt},$$

where $h_4(\epsilon) = -\epsilon \log_2(\epsilon/3) - (1-\epsilon)\log_2(1-\epsilon)$ is the quaternary entropy. At $\lambda = 5$, $\epsilon_s = 0.01$: $C \approx (1 - e^{-5})(2 - 0.07) \approx 1.91$ bits/nt.

The Shotgun DNA Sequencing Problem

The *shotgun sequencing problem* (in computational genomics) is closely related: given L/l reads of length l (where L is the genome length) covering the genome at depth $\lambda = L/l \cdot l/L =$ expected depth, reconstruct the genome. The Shannon-theoretic threshold for successful reconstruction is $\lambda \geq 1 + \varepsilon$ (for any $\varepsilon > 0$) when the genome has no repeats, by a random

coverage argument. With repeats of length r: threshold rises to $\lambda \geq \log L / \log l + O(1)$, a much higher coverage requirement — explaining why assembling repetitive genomes (human: $\approx 45\%$ repetitive) requires long reads ($l \geq r$) or paired-end reads.

S.2 Proof of the Polarization Theorem

We prove that fraction $I(W)$ of synthetic channels $W_n^{(i)}$ have $Z(W_n^{(i)}) \le 2^{-n^\beta}$ for any $\beta < 1/2$, and fraction $1 - I(W)$ have $Z(W_n^{(i)}) \ge 1 - 2^{-n^\beta}$.

Step 1: Martingale construction. Define the process $Z_\ell = Z(W_{2^\ell}^{(I_\ell)})$ where $I_\ell \in \{1, \dots, 2^\ell\}$ is chosen uniformly at random. Then:

$$\mathbb{E}[Z_{\ell+1} | Z_\ell] = \frac{1}{2}(2Z_\ell - Z_\ell^2) + \frac{1}{2} Z_\ell^2 = Z_\ell. \tag{4}$$

So Z_ℓ is a *martingale* with respect to the natural filtration. By the martingale convergence theorem, $Z_\ell \to Z_\infty$ a.s. as $\ell \to \infty$, for some random variable $Z_\infty \in [0, 1]$.

Step 2: The limit is 0 or 1. Z_∞ is a fixed point of the recursion: $\mathbb{E}[Z_\infty] = \mathbb{E}[Z_0] = Z(W)$. The fixed points of $f(z) = 2z - z^2$ in $[0, 1]$ are 0 and 1 (check: $f(0) = 0$, $f(1) = 1$, and $f(z) > z$ for $z \in (0, 1)$). So $Z_\infty \in \{0, 1\}$ a.s. Since $\mathbb{E}[Z_\infty] = Z(W) \in (0, 1)$, we have $P(Z_\infty = 0) = 1 - Z(W) = I(W)$ and $P(Z_\infty = 1) = Z(W) = 1 - I(W)$. (Using $I(W) \approx 1 - Z(W)$ for BMS channels.)

Step 3: Rate of convergence. The process $Z_\ell(1 - Z_\ell)$ is a supermartingale: $\mathbb{E}[Z_{\ell+1}(1 - Z_{\ell+1}) | Z_\ell] \le Z_\ell(1 - Z_\ell)/2$. By induction: $\mathbb{E}[Z_\ell(1 - Z_\ell)] \le Z(W)(1 - Z(W))/2^\ell$. Markov's inequality gives: $P(Z_\ell \in [\delta, 1 - \delta]) \le Z(W)(1 - Z(W))/(\delta(1 - \delta)2^\ell)$, which decays exponentially in $\ell = \log_2 n$, giving $P(Z_n^{(I)} \in [\delta, 1 - \delta]) \le C/n^{1/(1-\beta)}$ for appropriate $\delta = 2^{-n^\beta}$. ∎

S.3 The Polyanskiy–Poor–Verdú (PPV) Meta-Converse

The PPV meta-converse provides the tightest known converse bound for the AWGN channel at finite block length. Define the *information spectrum* for a channel $P_{Y|X}$ and auxiliary distribution Q_Y:

$$i(x; y) = \log \frac{P_{Y|X}(y|x)}{Q_Y(y)}.$$

The meta-converse lower-bounds the block error probability of any (n, M, ε) code:

Theorem — PPV Meta-Converse

For any input distribution P_X and any auxiliary distribution Q_Y:

$$\varepsilon \;\ge\; \mathbb{E}_{P_{X,Y}}\left[\mathbf{1}\left[i(X;Y) \le \log M - \log \frac{1}{\beta_{1-\varepsilon}(P_{XY}, P_X \otimes Q_Y)}\right]\right],$$

where $\beta_\alpha(P, Q)$ is the minimum type-II error probability of a binary hypothesis test between P and Q at type-I error $\le 1 - \alpha$.

For the AWGN channel with optimal Gaussian input and $Q_Y = P_Y$ (capacity-achieving output marginal), the meta-converse gives the *normal approximation*:

$$\log M^*(n, \varepsilon) = nC - \sqrt{nV}\, Q^{-1}(\varepsilon) + \frac{1}{2}\log n + O(1),$$

where $C = \frac{1}{2}\log_2(1 + \mathrm{SNR})$ is capacity and $V = \mathrm{SNR}(2 + \mathrm{SNR})/(1 + \mathrm{SNR})^2/(2 \ln 2)^2$ is the *channel dispersion*.

Numerical example. AWGN at SNR $= 0$ dB ($P/N = 1$): $C = 0.5$ bits/channel use, $V \approx 0.267$ (bits)^2/channel use. For $n = 100$ and $\varepsilon = 10^{-3}$:

$$R^* = 0.5 - \sqrt{0.267/100}\cdot Q^{-1}(10^{-3}) + \frac{\log 100}{2\cdot 100} = 0.5 - 0.0517\cdot 3.09 + 0.023 \approx 0.363 \text{ bits/channel use.}$$

This is 0.137 bits below capacity — the "third-generation" correction beyond the basic capacity formula.

S.4 Achievability via Random Codes: The Random Coding Union Bound

The *random coding union* (RCU) bound (Polyanskiy et al., 2010) gives a tighter achievability bound than the Shannon random coding exponent:

Theorem — RCU Bound

There exists an (n, M, ε) code with:

$$\varepsilon \;\le\; \mathbb{E}_{P_{X^n}}\!\left[\min\!\left(1,\; (M-1)\,P\!\left[i(X'^n;Y^n) \ge i(X^n;Y^n)\right]\right)\right],$$

where $(X^n, Y^n) \sim P_{X^n}P_{Y^n|X^n}$ is the original pair and $X'^n \sim P_{X^n}$ is an independent copy.

For Gaussian codebooks on the AWGN channel, the RCU bound equals (to leading order in the normal approximation) the achievability counterpart of the PPV converse, confirming that the normal approximation is tight to $O(1/\sqrt{n})$.

S.5 Proof of the Slepian–Wolf Theorem (Formal)

Random binning encoder. Generate 2^{nR_X} independent bins for X^n and 2^{nR_Y} bins for Y^n. Assign each $x^n \in \mathcal{X}^n$ independently and uniformly to one of the 2^{nR_X} bins; similarly for Y^n.

Decoder. Given bin indices (B_X, B_Y), output the unique (x^n, y^n) in the intersection $\text{Bin}_{B_X} \cap \text{Bin}_{B_Y}$ that is jointly ϵ-typical with the true pair, if such a unique pair exists.

Error analysis. The true pair (x^n, y^n) is typical with probability $\ge 1 - \epsilon_1(n)$. The probability of a "bad" triple $(x^n, \tilde{x}^n, y^n)$ where $\tilde{x}^n \neq x^n$ is in the same bin as x^n and $(x^n, y^n, \tilde{x}^n)$ are jointly typical:

$$\begin{aligned} P(\text{false triple}) &\le \sum_{\tilde{x}^n} 2^{-nR_X}\cdot P((\tilde{x}^n, y^n) \text{ jointly typical} | x^n, y^n) && (5)\\ &\le 2^{nH(X|Y)+n\epsilon}\cdot 2^{-nR_X}\cdot 2^{-n(I(X;Y)-\epsilon)} && (6)\\ &= 2^{n(H(X|Y)-R_X+\epsilon')}, && (7)\end{aligned}$$

which vanishes when $R_X > H(X|Y)$. Analogous bounds for Y^n and for the joint constraint. ∎

S.6 The Blahut–Arimoto Algorithm: Convergence Rate

The Blahut–Arimoto algorithm for channel capacity (alternating maximisation of $I(X;Y)$) converges at a linear rate with convergence factor $\rho^* \in (0,1)$.

Convergence factor. Let $I^* = C$ be the channel capacity and $I^{(t)}$ the mutual information at iteration t. Then:

$$C - I^{(t)} \;\le\; (C - I^{(0)})\cdot(\rho^*)^t,$$

where $\rho^* = 1 - \frac{\min_{xy} P^*_{XY}(x,y)}{P^*(X)P^*(Y)}$ is related to the minimum ratio of the optimal joint distribution to the product of marginals. For symmetric channels: $\rho^* \le 1 - 1/\log|\mathcal{X}|$, giving $O(\log|\mathcal{X}|)$ iterations for a prescribed accuracy. For the rate-distortion analogue, the factor is $\rho^* = 1 - P_{\min}/Z$ where $P_{\min}$ is the minimum source probability and Z the normalisation.

This linear convergence is not improvable in general (the algorithm is equivalent to EM on a missing-data log-likelihood, which converges at the same rate). However, recent accelerated variants using gradient information achieve superlinear convergence near the optimum.

Numerical Tour: Part 9 — Lattice Coding for the Gaussian Channel

Setup. 1D Gaussian channel $Y = X + Z$, $Z \sim \mathcal{N}(0,1)$, power constraint $P = 1$. Lattice $\Lambda_f = \frac{1}{\sqrt{3}}\mathbb{Z}$ (fine), $\Lambda_c = \sqrt{3}\mathbb{Z}$ (coarse). Nesting ratio: $V(\Lambda_c)/V(\Lambda_f) = 3/(\frac{1}{3}) = 9$, so $\log_2 9 = 3.17$ bits.

Encoding. Message $m \in \{0, 1, \dots, 8\}$ (encoding 3 bits) maps to coset representative $t_m = m/\sqrt{3}$. Dither $d \sim \text{Unif}[0, \sqrt{3})$. Transmitted: $x = [t_m - d] \bmod \sqrt{3}$, satisfying $\mathbb{E}[x^2] = (\sqrt{3})^2/12 = 0.25 < P = 1$. (This simplified example uses $P < 1$; optimal nesting adjusts Λ_f to use all available power.)

Decoding. Received $y = x + z$. Compute $Q_{\Lambda_f}(y + d)$: find nearest $\frac{k}{\sqrt{3}}$ to $y + d$. Reduce modulo Λ_c: $k \bmod 9$ gives $\hat{m}$.

Error probability. Error iff $|z| > \frac{1}{2\sqrt{3}} = 0.289$: $P_e = 2Q(0.289) = 2(1 - \Phi(0.289)) \approx 2 \times 0.386 = 0.77$. This is high because the example is for illustration, not optimised. With $n = 100$ dimensions, lattice point errors become $\le 2^{-n^\beta}$ by the polarization theorem.

Rate comparison. AWGN capacity at $P = 1$, $N = 1$: $C = \frac{1}{2}\log_2(2) = 0.5$ bits/channel use. The lattice code achieves 3.17 bits / (large n channel uses) ≈ 0.03 bits per channel use in this 1D simplified example — well below capacity, showing that the full n-dimensional construction is needed to approach C.

Numerical Tour: Part 10 — DNA Storage Rate Calculation

System parameters. Oligo length $L = 200$ nt; alphabet $\{A, C, G, T\}$; GC content constraint: only sequences with 40–60% GC content and no homopolymer run > 3 are valid.

Effective alphabet size after biochemical constraints. For $L = 200$: fraction of binary sequences satisfying GC balance and no run-3 constraint $\approx 0.62 \cdot 0.75 \approx 0.47$ (approximately, by CLT for GC content and Markov chain analysis for runs). So capacity per nt $\approx 1 + \log_2(0.47) \approx 0.925$ bits/nt from constraints alone.

With error correction overhead. Use RS$(n = 200, k = 160, d = 41)$ over $\mathbb{F}_4$: rate 0.8, correcting up to 20 substitutions ($\epsilon_s = 0.01$, so expected errors per oligo $= 2$, well within correction capability). Net information per oligo: $160 \log_2 4 \times 0.925 \approx 296$ bits.

Coverage for erasure protection. With $\lambda = 10$ coverage: $P(\text{oligo lost}) = e^{-10} \approx 4.5 \times 10^{-5}$. For $M = 10^5$ oligos, expected lost ≈ 4.5. Use outer RS$(M = 10^5, k = 9.95 \times 10^4, d = 51)$ for erasure correction. Effective rate: 99.5% of information oligos carry data. Total: $0.995 \times 296 \approx 295$ bits/oligo, or $295/200 \approx 1.47$ bits/nt.

Comparison with theoretical limit. At $\lambda = 10$, $\epsilon_s = 0.01$: $C_{\text{DNA}} \approx (1 - e^{-10})(2 - h_4(0.01)) \approx 1.93$ bits/nt. Our scheme achieves $1.47/1.93 \approx 76\%$ of the information-theoretic limit.

Numerical Tour: Part 11 — Weight Enumerator Bounds via LP

Delsarte LP bound. For $n = 23$ (Golay code), the LP bound gives $A_2(23, 7) \leq 2048 = 2^{11}$. The Golay code achieves this. Verification via Krawtchouk polynomials $K_i^{23}(x)$: the feasibility LP finds dual variables $f = (f_0, \ldots, f_{23})$ with $\sum_j f_j K_j^{23}(x) \leq 0$ for all $x \in \{1, \ldots, 23\}$ and $f_0 = 1$, giving $A \leq f_0 / \sum_j f_j(-K_j^{23}(0)) = 2048$.

Numerically. The optimal LP solution has $f_j = 0$ for $j \notin \{0, 7, 8, 11, 12, 15, 16, 23\}$, with $f_7 = f_8 = f_{11} = f_{12} = f_{15} = f_{16} = f_{23} = 1$ (the weight distribution of the dual — the Golay code is self-dual after extension). The LP is tight because the Golay code achieves $A_2(23, 7) = 2048$ exactly.

Formula Reference: Parts 6–11 Supplement

Quantity	Formula	Reference
Bhattacharyya param.	$Z(W) = \sum_y \sqrt{W(y\|0)W(y\|1)}$	Sec. 6.1
Polarisation: W^-	$Z(W^-) \leq 2Z(W) - Z(W)^2$	Sec. 6.1
Polarisation: W^+	$Z(W^+) = Z(W)^2$	Sec. 6.1
SC block error prob.	$P_{\text{block}} \leq \sum_{i \in \mathcal{A}} Z(W_n^{(i)})$	Sec. 6.3
Slepian-Wolf region	$R_X \geq H(X\|Y)$, $R_Y \geq H(Y\|X)$, $R_X + R_Y \geq H(X, Y)$	Sec. 7.1
Wyner-Ziv rate	$R^{WZ}(D) = \min_{p(u\|x)} I(X; U\|Y)$	Sec. 7.2
Gaussian WZ	$R^{WZ}(D) = \frac{1}{2} \log^+(\sigma^2_{X\|Y}/D)$	Sec. 7.2
AWGN capacity	$C = \frac{1}{2} \log_2(1 + P/N)$	Sec. 8.1
Water-filling	$P_i = (\nu - N_i)^+$, $\sum P_i = P$	Sec. 8.2
MAC capacity	$\sum_{j \in \mathcal{S}} R_j \leq I(X_{\mathcal{S}}; Y\|X_{\mathcal{S}^c})$	Sec. 8.3
Normal approximation	$\log M^* = nC - \sqrt{nV}Q^{-1}(\varepsilon) + O(\log n)$	Sec. 9.1
Dispersion (AWGN)	$V = \text{SNR}(2 + \text{SNR})/(1 + \text{SNR})^2/(2 \ln 2)^2$	Sec. 9.1
Lattice packing	$\delta(\Lambda) = r_{\text{pack}}^n \pi^{n/2}/(\Gamma(n/2 + 1)V(\Lambda))$	Sec. 10.1
Lattice NSM	$G(\Lambda) = D/V(\Lambda)^{2/n}$	Sec. 10.1
WOM capacity ($t = 2$)	$R_{WOM} = h(1/3) + h(1/2) \approx 1.918$ bits/cell	Sec. 11.1
Kendall distance	$d_K(\pi, \sigma) =$ min adjacent transpositions $\pi \to \sigma$	Sec. 11.1
VT code	$\text{VT}_a(n) = \{x : \sum_i i x_i \equiv a \pmod{n+1}\}$	Sec. 11.3
DNA channel capacity	$C_{DNA} \approx (1 - e^{-\lambda})(2 - h_4(\epsilon_s))$	Sec. 11.4

I Asymptotic Analysis: The Random Coding Exponent

I.1 The Error Exponent

For a DMC $P_{Y|X}$ and i.i.d. random codebook with input distribution P_X, the block error probability decays as:

$$P_e(n, 2^{nR}) \;\le\; e^{-nE_r(R,P_X)},$$

where $E_r(R, P_X) = \max_{0\le\rho\le 1}[E_0(\rho, P_X) - \rho R]$ is the random coding exponent, with

$$E_0(\rho, P_X) = -\log \sum_y \left(\sum_x P_X(x)\, P_{Y|X}(y|x)^{1/(1+\rho)} \right)^{1+\rho}.$$

At $\rho = 1$: $E_r = E_0(1, P_X) - R$ (Gallager bound); at $R = C$: $E_r = 0$.

I.2 The Expurgated Exponent

For rates near zero, the random coding exponent can be improved by expurgation:

$$E_{\text{ex}}(R, P_X) = \max_{0\le s\le 1} \max_{\rho} \left[E_0(\rho, P_X)/\rho - R\right],$$

achieved by removing the half of codewords with the most confusable pairs. The expurgated exponent $E_{\text{ex}}(R) \ge E_r(R)$ for all R and is strictly larger for $R < E_r^{-1}(0)$.

I.3 The Sphere-Packing (Converse) Exponent

A matching converse: for any sequence of codes with rate R and block error probability $P_e(n) \le e^{-nE}$:

$$E \;\le\; E_{\text{sp}}(R) \;=\; \max_{s\ge 0}[E_0(s/(1+s), P_X^*) - sR],$$

where P_X^* is the capacity-achieving input distribution. The sphere-packing exponent E_{sp} bounds the best achievable error exponent from above. For most rates $R \in (0, C)$, the random coding exponent $E_r(R)$ is strictly below $E_{\text{sp}}(R)$, indicating a gap between achievability and converse in the error exponent regime.

J Proofs of Selected Results from Parts 6–11

J.1 Proof of the Wyner-Ziv Rate for Gaussian Sources

Let $X \sim \mathcal{N}(0, \sigma^2)$, $Y = X + N$ with $N \sim \mathcal{N}(0, N_0)$ independent of X. The optimal auxiliary variable for the Wyner-Ziv problem is $U = X + \tilde{N}$ where $\tilde{N} \sim \mathcal{N}(0, D)$ independent of everything, and the decoder uses $\hat{X} = \mathbb{E}[X|U, Y]$ (MMSE estimator).

Compute: $\sigma^2_{X|Y} = \sigma^2 N_0/(\sigma^2+N_0)$ (standard Gaussian conditioning). The rate: $I(X;U|Y) = h(U|Y) - h(U|X,Y) = h(U|Y) - h(\tilde{N})$. $U|Y = (\mathbb{E}[X|Y] + \tilde{N}) + (\text{residual})$; since $X = \mathbb{E}[X|Y] + (X - \mathbb{E}[X|Y])$: $U = X + \tilde{N} = \mathbb{E}[X|Y] + (X - \mathbb{E}[X|Y]) + \tilde{N}$, so $U|Y \sim \mathcal{N}(\mathbb{E}[X|Y], \sigma^2_{X|Y} + D)$. Thus $h(U|Y) = \frac{1}{2}\log(2\pi e(\sigma^2_{X|Y} + D))$ and $h(\tilde{N}) = \frac{1}{2}\log(2\pi e D)$. So:

$$R^{WZ}(D) = I(X;U|Y) = \frac{1}{2}\log\frac{\sigma^2_{X|Y} + D}{D} = \frac{1}{2}\log\frac{\sigma^2_{X|Y}}{D} + \frac{1}{2}\log\!\left(1 + D/\sigma^2_{X|Y}\right).$$

The second term is non-negative; minimising over $D \le \sigma^2_{X|Y}$ and using the achievability of the Gaussian rate-distortion function gives $R^{WZ}(D) = \frac{1}{2}\log(\sigma^2_{X|Y}/D) = R_{X|Y}(D)$. ∎

J.2 LT Code Degree Distribution Optimality

LT codes (Luby, 2002) use a fountain encoder: each output symbol is the XOR of d input symbols, where d is drawn from a degree distribution $\Omega(d)$. The *ideal Soliton* distribution $\Omega^{\text{IS}}(d) = 1/k$ for $d = 1$ and $\Omega^{\text{IS}}(d) = 1/(d(d-1))$ for $d = 2, \dots, k$, designed so that exactly one symbol has degree 1 at each decoding step (peeling decoder). However, the ideal Soliton is numerically unstable (runs out of degree-1 nodes with high probability). The *robust Soliton* adds a correction:

$$\Omega^{\text{RS}}(d) = (\Omega^{\text{IS}}(d) + \tau(d))/Z,$$

where $\tau(d) = R/(dk)$ for $d = 1, \dots, k/R - 1$, $\tau(k/R) = R\ln(R/\delta)/k$, and Z is a normalisation constant, with $R = c\sqrt{k}\ln(k/\delta)$ for constant c. With the robust Soliton, the LT decoder recovers all k input symbols from $k(1 + O(1/\sqrt{k}))$ output symbols with probability $\geq 1 - \delta$.

Part 12 Coded Caching and Index Coding

Sections 12.1 – 12.3. The index coding problem: background information and broadcast; coded caching: the Maddah-Ali–Niesen scheme and the caching gain; connection to interference alignment and degrees of freedom; the capacity of coded caching networks; connections to network coding.

12.1 The Index Coding Problem

Problem Formulation

In the *index coding* problem (Bar-Yossef et al., 2011), a server holds n messages $x_1, \dots, x_n \in \{0,1\}^L$. There are n receivers; receiver i wants message x_i and has a subset $\mathcal{K}_i \subseteq \{x_j : j \neq i\}$ of other messages as *side information*. The server broadcasts a binary string $\mathcal{E}(x_1, \dots, x_n)$ of minimum length such that every receiver can decode its desired message.

The *broadcast rate* $\beta(G)$ of a directed side-information graph G (with an edge $i \to j$ if receiver j has message x_i as side information) satisfies:

$$\beta(G) \;=\; \frac{1}{\alpha(G)},$$

where $\alpha(G)$ is the *independence number* of the *confusion graph* $\bar{G}$. This elegant result connects index coding to graph theory.

Example — Index Coding: 3 Receivers

Server has $x_1, x_2, x_3 \in \mathbb{F}_2$. Receiver 1 wants x_1, knows x_2. Receiver 2 wants x_2, knows x_3. Receiver 3 wants x_3, knows x_1.
Naïve approach: broadcast (x_1, x_2, x_3): 3 bits.
Coded approach: broadcast $(x_1 \oplus x_2, x_2 \oplus x_3)$: 2 bits.

- Receiver 1: $x_1 = (x_1 \oplus x_2) \oplus x_2$ (uses side info x_2). ✓
- Receiver 2: $x_2 = (x_2 \oplus x_3) \oplus x_3$ (uses side info x_3). ✓
- Receiver 3: $x_3 = (x_1 \oplus x_2) \oplus (x_2 \oplus x_3) \oplus x_1$ (uses side info x_1). ✓

Coding gain: 1 bit saved (33% reduction).

Connection to Rank and Graph Theory

The *minrank* of a graph G over field $\mathbb{F}_q$ is:

$$\operatorname{minrk}_q(G) \;=\; \min\{\operatorname{rank}(A) : A \in \mathbb{F}_q^{n\times n},\; A_{ii} \neq 0,\; A_{ij} = 0 \text{ if } (i,j) \notin E(G)\}.$$

For linear index codes: $\beta_{\text{linear}}(G) = \operatorname{minrk}_{\mathbb{F}_2}(\bar{G})$ (the minrank of the complement graph over $\mathbb{F}_2$). The overall (non-linear) broadcast rate satisfies $\alpha(\bar{G}) \leq \beta(G) \leq \operatorname{minrk}(\bar{G})$, with equality for perfect graphs (clique cover number = chromatic number of complement).

Shannon capacity of graphs. The Shannon capacity $\Theta(G) = \lim_{n\to\infty} \alpha(G^n)^{1/n}$ (where G^n is the n-th strong product) measures the asymptotic independence number. For the Petersen graph, $\Theta \approx 4$ (exact value unknown), and the connection to index coding broadcasts is: the broadcast rate over n instances of the problem is $\beta(G^n)/n \to 1/\Theta(\bar{G})$.

12.2 Coded Caching: The Maddah-Ali–Niesen Scheme

The Coded Caching Model

A *coded caching system* (Maddah-Ali and Niesen, 2014) consists of:

- A server with N files $W_1, \dots, W_N$, each of F bits.
- K users, each with a local cache of size M files (i.e., MF bits).
- Two phases: **placement** (cache filling, before requests are known) and **delivery** (broadcast after requests are revealed).

The *caching gain* is the factor by which coded delivery reduces the broadcast load compared to uncoded delivery.

Maddah-Ali–Niesen Scheme

In the placement phase: each file W_n is split into $\binom{K}{KM/N}$ sub-files, and each user caches the sub-files corresponding to subsets of users that include themselves.
In the delivery phase: for each subset $\mathcal{S}$ of $KM/N + 1$ users, the server broadcasts $\bigoplus_{i \in \mathcal{S}} W_{d_i, \mathcal{S}\setminus\{i\}}$ where $W_{d_i,\mathcal{T}}$ is the sub-file of the i-th user's requested file W_{d_i} that is cached by users in $\mathcal{T}$.
Each user in $\mathcal{S}$ can decode its desired sub-file because it caches all other KM/N XOR terms. The total delivery load is:

$$R^*(t) = \frac{K(1 - t/N)}{1 + t} \quad \text{(files per broadcast signal)},$$

where $t = KM/N$ is the aggregate cache size in units of one file.

Interpretation. The denominator $(1 + t)$ is the *coded caching gain*: by multicasting to $1 + t$ users simultaneously via XOR, each user gets a useful XOR term (thanks to the structured cache placement). The numerator $K(1 - t/N)$ accounts for the fraction of each file not in the caches.

Example — Coded Caching: $K = 3$ Users, $N = 3$ Files, $M = 1$

$t = KM/N = 1$. Each file split into $\binom{3}{1} = 3$ sub-files: $W_n = (W_{n,1}, W_{n,2}, W_{n,3})$, each of $F/3$ bits. Cache of user 1: $\{W_{1,1}, W_{2,1}, W_{3,1}\}$ (first sub-file of each file). Cache of user 2: $\{W_{1,2}, W_{2,2}, W_{3,2}\}$. Cache of user 3: $\{W_{1,3}, W_{2,3}, W_{3,3}\}$.
Requests: user 1 wants W_1, user 2 wants W_2, user 3 wants W_3.
Delivery (for subset $\mathcal{S} = \{1, 2, 3\}$):

$$W_{1,\{2,3\}} \oplus W_{2,\{1,3\}} \oplus W_{3,\{1,2\}}.$$

User 1 knows $W_{2,\{1,3\}}$ (in cache at user 1? No — user 1's cache has sub-file indexed by $\{1\}$, i.e. $W_{n,1}$, not $W_{n,\{1,3\}}$). Let us re-index: user i caches $W_{n,\mathcal{T}}$ for all $\mathcal{T} \ni i$. So user 1 caches $W_{n,\{1\}}, W_{n,\{1,2\}}, W_{n,\{1,3\}}$ for each n.
With $t = 1$ ($M = 1, N = 3$): user 1 caches sub-files for the subsets of size 1 containing user 1: just $\{1\}$. So the XOR broadcast for pair $\{1, 2\}$ is $W_{1,2} \oplus W_{2,1}$. User 1 knows $W_{2,1}$, recovers $W_{1,2}$. User 2 knows $W_{1,2}$, recovers $W_{2,1}$. Total: $\binom{3}{2} = 3$ transmissions of $F/3$ bits each $= F$ bits total. Uncoded: 2 missing sub-files per user $\times$ 3 users = 6 transmissions of $F/3$ bits. Coded caching gain: $6/3 = 2\times$ reduction.

Information-Theoretic Optimality

Theorem — Optimality of MAN Scheme (Yu et al., 2018)

For the coded caching problem with $N \geq K$ and $M = tN/K$ for integer $t \in \{0, 1, \ldots, K\}$, the MAN delivery load $R^*(t) = K(1 - t/N)/(1 + t)$ is *information-theoretically optimal* for any uncoded placement scheme.

The converse is proved via a "cut-set" style argument: consider any subset S of s users and the s files they request. Each user has cached at most M files of information, so collectively the s users hold at most $\min(sM, NF)$ bits of information from the N files. The broadcast must convey the remaining bits, giving a lower bound on delivery load. Optimising over all subsets gives the exact expression $R^*(t)$.

For general (non-integer) t: $R^*(M/N \cdot K) = K(1-M/N)/(1+KM/N)$, achieved by memory sharing between adjacent integer points.

12.3 Connection to Interference Alignment

The MAN delivery phase is an instance of *index coding* (the sub-files of different users' requested files are mixed via XOR), which in turn is related to *interference alignment* in wireless networks.

Degrees of freedom. In a K-user interference channel (IC), the *degrees of freedom* (DoF) per user is $1/2$ (each user gets half the bandwidth, the IC suffers $1/2$ DoF loss per user). Maddah-Ali and Tse (2008) showed that with *global channel state information at transmitters* (CSIT), the K-user IC achieves $K/2$ total DoF — i.e., full interference alignment. The achievability uses precoding matrices designed to align all interference signals at each receiver into a reduced-dimensional subspace.

Topological interference management. When CSIT is not available but the topology of the IC is known (which transmitter-receiver pairs are connected), the DoF problem reduces exactly to an index coding problem over the IC topology graph. The optimal DoF is $1/\beta(G)$ where G is the conflict graph of the IC. This remarkable connection shows that information theory, coding theory, and graph theory converge in the design of wireless interference networks.

Sections 13.1 – 13.4. Qubits, quantum gates, and the Bloch sphere; quantum channels and Kraus operators; quantum entropy and von Neumann entropy; quantum error correction: the stabiliser formalism, CSS codes, and the $[[7,1,3]]$ Steane code; the quantum Shannon theorem; quantum capacity and degradable channels.

13.1 Qubits and Quantum Channels

Quantum States

A *qubit* is a unit vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \in \mathbb{C}^2$ with $|\alpha|^2 + |\beta|^2 = 1$. A system of n qubits lives in $(\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$. A *mixed state* is described by a density matrix $\rho \in \mathbb{C}^{d\times d}$ with $\rho \succeq 0$, $\operatorname{tr}(\rho) = 1$.

The Bloch sphere. A single-qubit density matrix is $\rho = \frac{1}{2}(\mathbf{I} + \mathbf{r}\cdot\boldsymbol{\sigma})$ where $\mathbf{r} \in \mathbb{R}^3$ with $|\mathbf{r}| \le 1$ is the Bloch vector and $\boldsymbol{\sigma} = (\sigma_X, \sigma_Y, \sigma_Z)$ are the Pauli matrices. Pure states correspond to $|\mathbf{r}| = 1$ (surface of Bloch sphere); mixed states to the interior.

Quantum Channels

A *quantum channel* is a completely positive trace-preserving (CPTP) map $\mathcal{E} : \mathcal{L}(\mathcal{H}_{\text{in}}) \to \mathcal{L}(\mathcal{H}_{\text{out}})$. By the Kraus representation theorem, every CPTP map has the form:

$$\mathcal{E}(\rho) = \sum_k K_k \rho K_k^\dagger, \qquad \sum_k K_k^\dagger K_k = \mathbf{I}.$$

The matrices $\{K_k\}$ are *Kraus operators*. Important examples:

- *Bit-flip channel:* $\mathcal{E}(\rho) = (1-p)\rho + pX\rho X$ (flips qubit with prob. p).
- *Phase-flip channel:* $\mathcal{E}(\rho) = (1-p)\rho + pZ\rho Z$ (flips phase with prob. p).
- *Depolarising channel:* $\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$.
- *Erasure channel:* $\mathcal{E}(\rho) = (1-p)\rho \otimes |0\rangle\langle 0| + p|e\rangle\langle e|$ (qubit erased with prob. p).

Von Neumann Entropy and Quantum Information Quantities

The *von Neumann entropy* of a density matrix ρ is:

$$S(\rho) = -\operatorname{tr}(\rho \log_2 \rho) = -\sum_i \lambda_i \log_2 \lambda_i,$$

where λ_i are the eigenvalues of ρ. Properties:

- $S(\rho) \ge 0$ with equality iff ρ is a pure state.
- $S(\rho) \le \log_2 d$ with equality iff $\rho = \mathbf{I}/d$ (maximally mixed state).
- *Concavity:* $S(\sum_i p_i \rho_i) \ge \sum_i p_i S(\rho_i)$.
- *Subadditivity:* $S(\rho_{AB}) \le S(\rho_A) + S(\rho_B)$, with equality iff $\rho_{AB} = \rho_A \otimes \rho_B$.
- *Strong subadditivity (SSA):* $S(\rho_{ABC}) + S(\rho_B) \le S(\rho_{AB}) + S(\rho_{BC})$. SSA is the most important inequality in quantum information theory.

The *quantum mutual information* is $I(A;B)_\rho = S(\rho_A) + S(\rho_B) - S(\rho_{AB}) \geq 0$, and the *quantum relative entropy* (quantum KL divergence) is $D(\rho\|\sigma) = \operatorname{tr}[\rho(\log_2 \rho - \log_2 \sigma)] \geq 0$ (Klein's inequality).

13.2 Quantum Error Correction: Stabiliser Codes

The Quantum Error Correction Conditions

Unlike classical errors (bit flips), quantum errors come in two types: *bit-flip* (X error) and *phase-flip* (Z error). Because quantum states can be superpositions, an error on a qubit can be a continuous rotation. However, the *digitisation of errors* (Knill–Laflamme, 1997) shows that correcting X and Z errors suffices to correct arbitrary errors.

Theorem — Knill–Laflamme Error Correction Conditions

A code with code space $\mathcal{C}$ and projector $P_\mathcal{C}$ corrects a set of errors $\{E_k\}$ iff for all k, l:

$$P_\mathcal{C} E_k^\dagger E_l P_\mathcal{C} = c_{kl} P_\mathcal{C},$$

for some Hermitian matrix $[c_{kl}]$. (The "non-deformability" condition: different error syndromes are orthogonal and errors do not mix within the code space.)

Stabiliser Codes

A *stabiliser code* $\mathcal{C}$ on n qubits is defined by an abelian group $\mathcal{S}$ of n-qubit Pauli operators ($\{I, X, Y, Z\}^{\otimes n}$ with signs ± 1) that do not contain $-\mathbf{I}$. The code space is:

$$\mathcal{C} = \{|\psi\rangle : g|\psi\rangle = |\psi\rangle \text{ for all } g \in \mathcal{S}\}.$$

For a stabiliser group generated by $n-k$ independent generators $g_1, \ldots, g_{n-k}$, the code encodes k logical qubits in n physical qubits: a $[[n, k, d]]$ code.

Error detection. Measuring the syndrome $(s_1, \ldots, s_{n-k})$ where $s_i = \pm 1$ is the eigenvalue of g_i on the received state determines which error (or error pattern) occurred, without disturbing the logical information — exactly as syndrome decoding for classical linear codes, but in the Heisenberg picture.

The $[[7, 1, 3]]$ Steane Code

The *Steane code* encodes 1 logical qubit in 7 physical qubits, with minimum distance 3 (correcting one error). It is a CSS code derived from the classical $[7, 4, 3]$ Hamming code.

Stabiliser generators (6 generators for $n - k = 7 - 1 = 6$):

$$\begin{aligned}
g_1 &= I\,I\,I\,X\,X\,X\,X \\
g_2 &= I\,X\,X\,I\,I\,X\,X \\
g_3 &= X\,I\,X\,I\,X\,I\,X \\
g_4 &= I\,I\,I\,Z\,Z\,Z\,Z \\
g_5 &= I\,Z\,Z\,I\,I\,Z\,Z \\
g_6 &= Z\,I\,Z\,I\,Z\,I\,Z
\end{aligned}$$

(Here X means σ_X, Z means σ_Z, I means identity on that qubit.) The X-type stabilisers g_1, g_2, g_3 detect Z errors; the Z-type stabilisers detect X errors. The parity-check matrix of the underlying $[7, 4, 3]$ Hamming code appears in both halves: this is the CSS structure.

Logical operators. The logical $\bar{X}$ and $\bar{Z}$ operators (which act on the encoded qubit without being detected by the syndrome) are:

$$\bar{X} = X^{\otimes 7} (\text{product of all } X), \quad \bar{Z} = Z^{\otimes 7}.$$

These have weight 7, confirming $d = 3$ (any Pauli error of weight ≤ 1 is detectable and correctable).

13.3 The Quantum Shannon Theorem

Quantum Channel Capacity

The *quantum capacity* $Q(\mathcal{E})$ of a quantum channel $\mathcal{E}$ is the maximum rate (in qubits per channel use) at which quantum information can be transmitted reliably. The quantum Shannon theorem:

Theorem — Quantum Channel Capacity (LSD Theorem)

$$Q(\mathcal{E}) = \lim_{n \to \infty} \frac{1}{n} I_c(\mathcal{E}^{\otimes n}),$$

where the *coherent information* is $I_c(\mathcal{E}) = \max_{\rho_A} [S(\mathcal{E}(\rho_A)) - S_e(\rho_A, \mathcal{E})]$, with S_e the *entropy exchange* (entropy of the environment after the channel acts).

For *degradable channels* (where the channel to the environment is a degraded version of the channel to the receiver), the regularisation is not needed: $Q(\mathcal{E}) = I_c(\mathcal{E})$. The erasure channel and amplitude damping channel are degradable.

Example — Quantum Erasure Channel Capacity

The quantum erasure channel with erasure probability p: $\mathcal{E}(\rho) = (1-p)\rho + p|e\rangle\langle e|$. Coherent information:

$$I_c = (1-p)\log_2 d - H(p, 1-p) = (1-p)\log_2 d - h(p).$$

For a qubit ($d = 2$): $Q = (1 - 2p)$ for $p \leq 1/2$, zero otherwise. Compare with classical erasure capacity $C = 1 - p$ — quantum capacity is strictly smaller because quantum erasures destroy correlations.

Superdense Coding and Quantum Teleportation

Two foundational protocols connect quantum and classical information:

Superdense coding. Using one pre-shared EPR pair ($|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$) and one qubit channel, transmit 2 classical bits: Alice applies one of $\{I, X, Z, XZ\}$ to her qubit (encoding 2 bits), sends it to Bob. Bob measures the Bell basis. This doubles the classical capacity of a qubit channel — the pre-shared entanglement acts as a resource doubling the rate. Formally: $C_E(\mathcal{E}) = 2C(\mathcal{E})$ for an ideal qubit channel $\mathcal{E}$.

Quantum teleportation. Transmit one qubit using one pre-shared EPR pair and 2 classical bits: Alice makes a Bell measurement on (her EPR qubit, the qubit to teleport), sends 2 classical bits to Bob. Bob applies the appropriate Pauli correction to his EPR qubit, recovering the teleported state. Formally: $Q_E(\mathcal{E}) = C(\mathcal{E})/2$ for a classical bit channel $\mathcal{E}$ — each classical bit provides half a qubit of quantum communication when supplemented by en-

tanglement. Together, superdense coding and teleportation exhibit the *unit resource capacity region* of quantum channels.

13.4 The Hashing Bound and Stabiliser Code Design

The Quantum Hamming Bound

For a $[[n, k, d]]$ stabiliser code (correcting $t = \lfloor (d-1)/2 \rfloor$ errors):

$$\sum_{j=0}^{t} \binom{n}{j} 3^j \;\leq\; 2^{n-k}.$$

The factor 3^j counts the X, Y, Z error patterns of weight j (three types per qubit). A code achieving this bound is called *quantum perfect* (or *nondegenerate*).

The *quantum Singleton bound*: $k \leq n - 4(d-1)$ for pure codes ($d \geq 1$). Codes achieving this are quantum MDS (QMDS) codes. Known QMDS codes exist for small parameters (e.g., $[[5, 1, 3]]$ code achieves $1 = 5 - 4(2) = 5 - 8$? No — for $d = 3$: $k \leq 5 - 8 < 0$, so the $[[5, 1, 3]]$ code is *not* QMDS. In fact the quantum Singleton bound gives $k \leq n - 4(d-1)$, so for $n = 5, d = 3$: $k \leq 5 - 8 = -3$, impossible. The $[[5, 1, 3]]$ code is instead optimal in the sense of the quantum Hamming bound: $\binom{5}{0} + 3\binom{5}{1} = 1 + 15 = 16 \leq 2^{5-1} = 16$, achieved with equality — it is a perfect code.)

Degenerate vs. Non-Degenerate Codes

A crucial difference from classical coding: quantum codes can be *degenerate* — two distinct errors may have the same syndrome and thus be indistinguishable, yet both correct without error because they act identically within the code space. This has no classical analogue. Degenerate codes can potentially achieve higher rates than the quantum Hamming bound suggests for non-degenerate codes, and understanding when degeneracy helps is an open problem in quantum coding theory.

Numerical Tour: Part 12 — Coded Caching Gain Calculation

System: $K = 12$ users, $N = 4$ files, $M = 1$ file per cache ($t = KM/N = 3$).

MAN delivery load: $R^*(3) = \frac{12(1-1/4)}{1+3} = \frac{12\times 0.75}{4} = 2.25$ files.

Uncoded (naive) delivery: Each user requests a file; at most 12 distinct requests, but only 4 files. By the union bound, at most 4 distinct files needed. Uncoded: $4(1-1/4) = 3$ files (each file partially cached). So uncoded delivery load $= 3$ files.

Coded caching gain: $3/2.25 = 4/3 \approx 1.33\times$. This gain equals $1 + t = 4$.

As $K \to \infty$ with M/N fixed: $t = KM/N \to \infty$, $R^*(t) \to K(1-M/N)/(KM/N) = N/M-1$. The load per file becomes $N/M-1$, independent of K — a *global caching gain* that scales with K (more users = more multicast opportunities = lower per-user load).

Numerical Tour: Part 13 — Steane Code Error Correction

Error model: single-qubit depolarising with $p = 0.01$ per qubit, applied to all 7 physical qubits of the Steane code.

Syndrome measurement. For each stabiliser g_i, measure eigenvalue $s_i = \pm 1$. With one X error on qubit j: Z-type stabilisers g_4, g_5, g_6 give syndromes s_4, s_5, s_6 encoding the binary representation of j (since the parity-check matrix of the Hamming code has columns 001 through 111). Similarly for Z errors and X-type stabilisers.

Error probability before correction: $P(\geq 1$ qubit error on 7 qubits$) \leq 7p = 0.07$.

After correction (corrects any single-qubit error): $P(\text{residual error after correction}) \leq \binom{7}{2}p^2 \approx 21 \times 10^{-4} = 0.0021$. Correcting single-qubit errors reduces the logical error rate from 7% to $\approx 0.21\%$ — a $33\times$ improvement. At $p = 0.001$: improvement is $\approx 4000\times$.

Threshold. At the *fault-tolerance threshold* p_{th}: the logical error rate equals the physical error rate. For the Steane code: $21p_{\text{th}}^2 = p_{\text{th}}$ gives $p_{\text{th}} \approx 1/21 \approx 4.8\%$. For concatenated Steane codes, the threshold is near $p_{\text{th}} \approx 1\%$ (accounting for measurement errors). Surface codes achieve higher thresholds ($\approx 1\%$ for depolarising noise) at the cost of higher overhead.

Formula Reference: Parts 12–13

Quantity	Formula	Reference
Index coding rate	$\beta(G) = 1/\alpha(\bar{G})$	Sec. 12.1
Linear index code	$\beta_{\text{lin}}(G) = \text{minrk}(\bar{G})$	Sec. 12.1
MAN delivery load	$R^*(t) = K(1-t/N)/(1+t)$	Sec. 12.2
Caching gain	$1 + t = 1 + KM/N$	Sec. 12.2
Von Neumann entropy	$S(\rho) = -\text{tr}(\rho \log_2 \rho)$	Sec. 13.1
Quantum mutual info.	$I(A;B) = S(\rho_A) + S(\rho_B) - S(\rho_{AB})$	Sec. 13.1
Quantum rel. entropy	$D(\rho\|\sigma) = \text{tr}[\rho(\log\rho - \log\sigma)]$	Sec. 13.1
Stabiliser code	$\mathcal{C} = \{\|\psi\rangle : g\|\psi\rangle = \|\psi\rangle \ \forall g \in \mathcal{S}\}$	Sec. 13.2
KL error correction	$P_{\mathcal{C}} E_k^\dagger E_l P_{\mathcal{C}} = c_{kl} P_{\mathcal{C}}$	Sec. 13.2
Quantum Hamming	$\sum_{j=0}^{t} \binom{n}{j} 3^j \leq 2^{n-k}$	Sec. 13.4
Quantum Singleton	$k \leq n - 4(d-1)$	Sec. 13.4
Quantum capacity	$Q(\mathcal{E}) = \lim_{n\to\infty} \frac{1}{n} I_c(\mathcal{E}^{\otimes n})$	Sec. 13.3
Coherent information	$I_c = S(\mathcal{E}(\rho)) - S_e(\rho, \mathcal{E})$	Sec. 13.3

Superdense coding	2 classical bits from 1 qubit + 1 ebit	Sec. 13.3
Teleportation	1 qubit from 2 classical bits + 1 ebit	Sec. 13.3

K The Information-Geometric Perspective on Codes and Algorithms

K.1 Exponential Families and Linear Codes

The *exponential family* generated by statistics $T : \mathcal{X} \to \mathbb{R}^k$ is:

$$p_\theta(x) = \exp[\theta^\top T(x) - A(\theta)]\mu(x),$$

where $A(\theta) = \log \sum_x \exp[\theta^\top T(x)]\mu(x)$ is the log-partition function and $\theta \in \mathbb{R}^k$ are the *natural parameters*. For binary linear codes: setting μ = counting measure on $\{0,1\}^n$, $T(x) = Hx^\top$ (syndrome), and θ determined by the channel LLRs, the posterior over codewords is exactly the exponential family with sufficient statistic T. This connection motivates:

- The syndrome $\mathbf{Hy}$ is the sufficient statistic for inferring the error.
- The optimal (MAP) decoder maximises $\theta^\top T(c)$ over $c \in \mathcal{C}$, which is linear in θ (the channel LLRs) — giving a linear programming relaxation of ML decoding.
- Iterative decoding (BP) is an approximate alternating minimisation of KL divergence on the exponential family manifold defined by the Tanner graph factor structure.

K.2 The e-Flat and m-Flat Submanifolds

In information geometry, the manifold of probability distributions has two natural coordinate systems: the *θ-coordinates* (natural parameters) and the *η-coordinates* (mean parameters $\eta = \mathbb{E}_\theta[T(X)]$). A submanifold is *$e$-flat* (exponentially flat) if it is an affine subspace in θ-coordinates, and *m-flat* (mixture-flat) if it is an affine subspace in η-coordinates.

For the turbo/LDPC message-passing decoder:

- The joint distribution of all bits in the Tanner graph defines a high-dimensional exponential family.
- The true posterior (constrained to codewords) is the e-projection onto the m-flat submanifold of distributions that factorise according to the Tanner graph.
- Belief propagation iteratively applies m-projections and e-projections (via the Pythagoras theorem of information geometry), converging to the correct posterior when the Tanner graph is a tree.

K.3 Pythagorean Theorem for KL Divergence

For e-flat and m-flat submanifolds $\mathcal{M}_e$ and $\mathcal{M}_m$ with a unique intersection point p^*:

$$D_{\mathrm{KL}}(p \,\|\, q) = D_{\mathrm{KL}}(p \,\|\, p^*) + D_{\mathrm{KL}}(p^* \,\|\, q) \quad \forall\, p \in \mathcal{M}_e, q \in \mathcal{M}_m.$$

This is the *information-geometric Pythagoras theorem*. Applied to coding:

- $\mathcal{M}_e$ = constraints from the channel (the posterior given the syndrome).
- $\mathcal{M}_m$ = constraints from the code structure (factorisable distributions).
- p^* = the true posterior = the unique point satisfying both constraints = the ML codeword.

The alternating projections of BP minimise the total "distance" (KL divergence) from both constraint sets simultaneously. This geometric view explains why BP works well on sparse

graphs: the projections are nearly orthogonal (Pythagoras theorem approximately holds) when the graph has large girth.

L Selected Problems with Solutions

L.1 Reed–Solomon MDS Property

Problem. Prove that the Reed–Solomon code RS(n, k) over $\mathbb{F}_q$ with $n = q - 1$ is MDS, i.e., achieves $d_{\min} = n - k + 1$.

Solution. We already know $d_{\min} \le n - k + 1$ (Singleton bound). It suffices to show $d_{\min} \ge n - k + 1$, i.e., every non-zero codeword has weight $\ge n - k + 1$.

A codeword is $c = (f(\alpha^0), \dots, f(\alpha^{n-1}))$ for a non-zero polynomial f of degree $\le k - 1$. A zero position in c corresponds to a root of f: the number of zeros in c equals the number of roots of f in $\{\alpha^0, \dots, \alpha^{n-1}\} = \mathbb{F}_q^*$. Since f has degree $\le k-1$, it has at most $k-1$ roots in any field. Therefore, the weight of c is $\ge n - (k-1) = n - k + 1$. ∎

L.2 Huffman Code Expected Length Bounds

Problem. For a source X with entropy $H(X)$, show that the expected length $\bar{L}$ of the Huffman code satisfies $\bar{L} < H(X) + 1$.

Solution. Assign length $l_i = \lceil -\log_2 p_i \rceil$ to symbol x_i. These satisfy Kraft's inequality: $\sum_i 2^{-l_i} \le \sum_i 2^{+\log_2 p_i} = \sum_i p_i = 1$. So a prefix-free code with these lengths exists, and the Huffman code has expected length $\bar{L} \le \sum_i p_i \lceil -\log_2 p_i \rceil < \sum_i p_i(-\log_2 p_i + 1) = H(X) + 1$. ∎

L.3 LDPC Density Evolution Fixed Point

Problem. For a $(3, 6)$-regular LDPC code on the BEC(ϵ), show the density evolution recursion is $p_\ell = \epsilon(1 - (1 - p_{\ell-1})^5)^2$ and find the threshold ϵ^*.

Solution. Variable-to-check message: p = erasure probability. Each variable has degree 3; a variable-to-check message is erased iff the channel bit is erased AND both other incoming check messages are erased. With check-to-variable erasure probability q: $p = \epsilon \cdot q^2$ (2 incoming check messages, each with erasure prob. q).

Each check has degree 6; a check-to-variable message is not erased iff all 5 other incoming variable messages are not erased: $q = 1 - (1 - p)^5$.

Fixed-point equation: $p = \epsilon(1 - (1 - p)^5)^2$. At $p = 0$: $0 = 0$ (trivial fixed point, successful decoding). The non-trivial fixed point $p^* > 0$ appears when $\frac{d}{dp}[\epsilon(1 - (1 - p)^5)^2]\big|_{p=0} \ge 1$, i.e., $10\epsilon \ge 1$, giving $\epsilon^* = 1/10 = 0.1$. For $\epsilon < 0.1$, decoding converges to $p = 0$; for $\epsilon > 0.1$, a non-trivial fixed point exists and decoding fails. ∎

L.4 The Gaussian Rate-Distortion Converse

Problem. Show that for $X \sim \mathcal{N}(0, \sigma^2)$ with squared-error distortion, no code with rate $R < \frac{1}{2}\log_2(\sigma^2/D)$ can achieve MSE $\le D$.

Solution. For any encoder-decoder pair, the Markov chain $X \to M \to \hat{X}$ gives $I(X; \hat{X}) \le I(X; M) \le \log_2 |\mathcal{M}| = nR$ (by data processing and rate constraint). The rate-distortion function lower bounds $I(X; \hat{X}) \ge R(D) = \frac{1}{2}\log_2(\sigma^2/D)$ whenever $\mathbb{E}[(X - \hat{X})^2] \le D$ (by defini-

tion of $R(D)$). Combining: $nR \geq I(X;\hat{X}) \geq R(D) = \frac{1}{2}\log_2(\sigma^2/D)$, so $R \geq \frac{1}{2n}\log_2(\sigma^2/D)$. For an i.i.d. block of length n: $R \geq \frac{1}{2}\log_2(\sigma^2/D)$ after dividing by n. ■

L.5 Kolmogorov Complexity: Non-Computability

Problem. Show directly (via the Berry paradox) that $K(x)$ is not computable.

Solution. Suppose for contradiction that there is a Turing machine T computing $K(x)$ for all x. Define the following algorithm A: enumerate all binary strings $x_1, x_2, \ldots$ in lexicographic order; for each x_i, compute $K(x_i)$; output the first x_i with $K(x_i) > 1000$.

Such a string must exist: only $\sum_{l=0}^{1000} 2^l < 2^{1001}$ strings have $K \leq 1000$.

The algorithm A is a valid Turing machine, so $K(\text{output of } A) \leq |A| + O(1) \leq c$ for some constant $c \leq 1000$ (the description of A is short).

But by construction, the output x satisfies $K(x) > 1000 > c$. Contradiction. ■

L.6 Channel Polarisation: BEC Example

Problem. For the BEC($\epsilon = 0.5$) with $n = 4$, compute the Bhattacharyya parameters of all 4 synthetic channels $W_4^{(i)}$, $i = 1, 2, 3, 4$, and determine the information set $\mathcal{A}$ for rate $R = 1/2$ (2 information bits).

Solution. Start with $Z = 0.5$. Level 1: $Z^- = 2(0.5) - (0.5)^2 = 0.75$, $Z^+ = (0.5)^2 = 0.25$. Level 2 from $Z^- = 0.75$: $Z^{--} = 2(0.75) - (0.75)^2 = 0.9375$, $Z^{-+} = (0.75)^2 = 0.5625$. Level 2 from $Z^+ = 0.25$: $Z^{+-} = 2(0.25) - (0.25)^2 = 0.4375$, $Z^{++} = (0.25)^2 = 0.0625$.

The bit-reversal permutation maps the recursion order to channels: $W_4^{(1)} \leftrightarrow Z^{--} = 0.9375$ (worst), $W_4^{(2)} \leftrightarrow Z^{-+} = 0.5625$, $W_4^{(3)} \leftrightarrow Z^{+-} = 0.4375$, $W_4^{(4)} \leftrightarrow Z^{++} = 0.0625$ (best).

For rate $R = 1/2$ (2 information bits, 2 frozen): select the 2 channels with smallest Z. $\mathcal{A} = \{3, 4\}$ (channels with $Z = 0.4375$ and 0.0625). ■

Closing Synthesis: Unifying Themes

Looking across the five parts and supplementary chapters, three unifying mathematical themes emerge.

Theme 1: Duality. The deepest results in coding and information theory come in dual pairs. Encoder/decoder; generator/parity-check; capacity/rate-distortion; θ/η coordinates in information geometry; e-flat/m-flat submanifolds; primal/dual of the Blahut–Arimoto algorithm; coherent/classical information. This duality is not a coincidence: it reflects the underlying symplectic structure of information theory, most explicitly visible in the quantum setting (where X and Z errors are truly dual) and in the geometric picture (where the Fisher metric is self-dual).

Theme 2: Polarisation and Extremisation. Optimal codes always arise from extremal constructions: Reed–Solomon codes extremise the Singleton bound; turbo and LDPC codes push towards the Shannon limit; polar codes achieve it by the polarisation phenomenon; Gaussian inputs extremise channel capacity and minimise the Wyner-Ziv rate. The mathematical engine is always an extremisation of mutual information, entropy, or KL divergence — often solved by Lagrange multipliers leading to exponential families (Gibbs distributions), and often implementable by iterative alternating-direction algorithms.

Theme 3: The Role of Randomness. Random constructions permeate the theory: random

codebooks (Shannon's capacity proof), random binning (Slepian–Wolf and Wyner–Ziv), random LDPC graphs (threshold analysis), random interleavers (turbo code analysis), random DNA pools (fountain codes for DNA storage). In each case, randomness proves existence of good codes without providing an explicit construction; the challenge of explicit construction (polynomial-time encodable and decodable codes achieving the random code performance) is the central engineering problem, now largely resolved for classical channels (polar codes, spatially coupled LDPC) but still open for quantum channels and multi-user networks.

Volume III will return to these themes through the lens of information geometry, where the Fisher information metric, geodesics, and divergences provide a differential-geometric framework unifying the statistical, informational, and algebraic structures that appear throughout Volumes I and II.

M Advanced Topics in Decoding Complexity

M.1 The Complexity of ML Decoding

Theorem — Berlekamp–McEliece–van Tilborg (1978)

Maximum-likelihood decoding of a general binary linear code is NP-hard.

Proof sketch. Reduce from the *Syndrome Decoding Problem* (SDP): given $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, $\mathbf{s} \in \mathbb{F}_2^r$, integer w, does there exist $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathbf{H}\mathbf{e}^\top = \mathbf{s}$ and $\mathrm{wt}(\mathbf{e}) \leq w$? SDP is NP-complete (it generalises 3-dimensional matching and set cover). ML decoding reduces to SDP (find minimum-weight coset leader), establishing NP-hardness.

Implications. Since ML decoding is NP-hard in general, practical decoders for long codes must be suboptimal. The art of code design is to engineer the code structure (linearity, sparsity, algebraic richness) so that efficient algorithms (Viterbi, Berlekamp–Massey, BP) achieve performance close to ML. For LDPC codes on the BEC, BP decoding is exactly ML for cycle-free graphs; for general channels and cyclic graphs, the gap (measured in SNR) is small for well-designed irregular LDPC codes (< 0.1 dB for long codes).

M.2 LP Decoding and the Fundamental Polytope

Linear programming (LP) decoding (Feldman et al., 2005) relaxes the ML decoding integer programme to a linear programme:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c} \in \mathcal{P}(H)} \langle \boldsymbol{\lambda}, \mathbf{c} \rangle,$$

where $\boldsymbol{\lambda}$ is the vector of channel LLRs and $\mathcal{P}(H)$ is the *fundamental polytope*: the convex hull of all codewords, relaxed to allow fractional solutions at check-node intersections. For a sparse code, this polytope has $O(n \cdot 2^{d_c})$ facets and the LP has $O(n d_c)$ variables, solvable in polynomial time.

LP decoding properties:

- *ML certificate:* if LP returns an integer solution, it is the ML codeword.
- *Pseudocodewords:* LP can return non-integer (fractional) solutions. The LP threshold is determined by the minimum pseudocodeword weight in the Tanner graph.
- *Threshold:* for regular LDPC codes on the AWGN, LP decoding has the same asymptotic threshold as BP decoding, but improved finite-length performance near the error floor.

M.3 GRAND: Universal Decoding

GRAND (Duffy et al., 2019) is a universal decoding algorithm: code-agnostic, it queries noise patterns $\mathbf{e}$ in decreasing likelihood order, checking $\mathbf{H}(\mathbf{y} - \mathbf{e})^\top = \mathbf{0}$ at each step and returning the first valid codeword. For the BSC(p), the expected number of queries is $2^{nh(p)}$, making GRAND practical for low-noise channels and short codes ($n \leq 128$) where hardware implementations achieve very high throughput for 5G and WiFi short-packet scenarios.

N Lossy Compression Standards: JPEG to Neural Codecs

N.1 JPEG and JPEG 2000

JPEG (1992) operates on 8×8 pixel blocks. After YCbCr conversion and chroma subsampling, each block undergoes the 2D-DCT:

$$F_{uv} = \frac{c_u c_v}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} f_{xy} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}.$$

Coefficients are divided by a quality-dependent quantisation matrix Q_{uv} and entropy coded (DPCM on DC, RLE + Huffman on AC). The rate-distortion curve follows approximate reverse water-filling over the 64 DCT coefficients, with each additional bit per pixel reducing distortion by ≈ 6 dB.

JPEG 2000 replaces the block DCT with a global wavelet transform (CDF 9/7 for lossy, LeGall 5/3 for lossless), gains 1–2 dB over JPEG, and supports embedded progressive bitstreams via EBCOT (embedded block coding with optimised truncation).

N.2 Video Coding: H.264/AVC and HEVC

Video coding adds temporal compression via motion-compensated prediction:

$$\hat{f}_t(x, y) = f_{t-k}(x + dx,\, y + dy)$$

for motion vector (dx, dy). The residual $r = f_t - \hat{f}_t$ has variance $\sigma_r^2 \approx \sigma^2(1 - \rho^2)$ where $\rho \approx 0.95$ for typical video, giving $\sigma_r^2 \approx 0.1\sigma^2$ — saving ≈ 1.65 bits/pixel at any distortion level. HEVC (H.265) uses flexible quadtree CU partitioning, 33 angular intra-prediction modes, and CABAC entropy coding, achieving $\approx 2\times$ bit-rate reduction over H.264 at the same quality.

N.3 Neural Image Compression

The standard neural codec architecture (Ballé et al., 2018) uses:

1. Analysis CNN f_a: $\mathbf{y} = f_a(\mathbf{x}; \phi)$.
2. Quantisation: $\hat{\mathbf{y}} = \text{round}(\mathbf{y})$.
3. Hyperprior entropy model: learned scale parameters $\hat{\sigma}$; rate $R = \mathbb{E}[-\log_2 p_{\hat{\mathbf{y}}}(\hat{\mathbf{y}}|\hat{\sigma})]$.
4. Synthesis CNN f_s: $\hat{\mathbf{x}} = f_s(\hat{\mathbf{y}}; \theta)$.

Training minimises $\mathbb{E}[R + \lambda D]$. On the Kodak benchmark, the joint autoregressive hyperprior (Minnen et al., 2018) achieves -42% BD-rate vs. JPEG and $+2.6$ dB PSNR at 0.5 bpp. The hyperprior is the neural analogue of Wyner–Ziv side information, reducing the gap between marginal and conditional entropy of the latent representation.

N.4 The Perception–Distortion Trade-off

Blau and Michaeli (2018) proved that no codec can simultaneously minimise MSE distortion and achieve perceptual quality ($p_{\hat{X}} = p_X$). The minimum distortion at perfect perceptual quality and rate R is bounded below by the MMSE reconstruction variance. This fundamental three-way trade-off (rate, distortion, perception) implies GAN-based codecs (which target $p_{\hat{X}} = p_X$) must sacrifice MSE fidelity — explaining why high-quality GAN reconstructions often score lower on PSNR despite looking better to human observers.

Final Formula Compendium

C.1 Algebraic Codes

Object	Formula / Property
$\mathbb{F}_q$ mult.	$\alpha^i \cdot \alpha^j = \alpha^{(i+j) \bmod (q-1)}$
Encoding	$\mathbf{c} = \mathbf{u}\mathbf{G}, \quad \mathbf{H}\mathbf{c}^\top = \mathbf{0}$
Syndrome	$\mathbf{s} = \mathbf{H}\mathbf{y}^\top = \mathbf{H}\mathbf{e}^\top$
Singleton bound	$d \le n - k + 1$
Hamming bound	$\sum_{i=0}^{t} \binom{n}{i}(q-1)^i \le q^{n-k}$
GV bound	$k/n \ge 1 - h_q(d/n)$
RS code	$[q-1, k, q-k]_q$; MDS; evaluation code
BCH bound	$d_{\min} \ge d_{\text{des.}}$ (Vandermonde)
MacWilliams	$W_{\mathcal{C}^\perp}(x,y) = \lvert\mathcal{C}\rvert^{-1} W_{\mathcal{C}}(x+y, x-y)$

C.2 Iterative Codes

Object	Formula / Property
Viterbi	$\alpha_t(s') = \min_s[\alpha_{t-1}(s) + \gamma_t(s,s')]$
BCJR LLR	$L^{\text{post}} = L^{\text{ch}} + L^{\text{ext}}$
BP var→chk	$\mu_{j\to i} = \lambda_j + \sum_{i' \ne i} \nu_{i'\to j}$
BP chk→var	$\nu_{i\to j} = 2\tanh^{-1} \prod_{j' \ne j} \tanh(\mu_{j'\to i}/2)$
Min-sum	$\nu = (\prod \text{sgn}) \cdot \min \lvert\mu\rvert$
LDPC rate	$R \ge 1 - \int \rho / \int \lambda$
BP threshold	$\sigma^* = \sup\{\sigma : \lim_l P_e^{(l)} = 0\}$
Polar recursion	$Z^- \le 2Z - Z^2, \quad Z^+ = Z^2$

C.3 Source Coding

Object	Formula / Property
Huffman	$H(X) \le \bar{L} < H(X) + 1$
Arithmetic	$L \le -\log_2 P(x^n) + 2$
LZ univ.	$L_{\text{LZ}}/n \to H$ a.s.
BWT LF	$\text{LF}(i) = C[L[i]] + \text{rank}_{L[i]}(i)$
CTW	$P_e(x^n \lvert s) = \frac{1}{2}\hat{P}_s + \frac{1}{2}P_e(\cdot\lvert 0s)P_e(\cdot\lvert 1s)$

C.4 Rate-Distortion

Object	Formula / Property
$R(D)$	$\min_{P_{\hat{X}\lvert X}:\mathbb{E}d \le D} I(X;\hat{X})$
Opt. test channel	$P^*(\hat{x}\lvert x) \propto P(\hat{x}) e^{-\lambda d(x,\hat{x})}$
Gaussian $R(D)$	$\frac{1}{2}\log_2^+(\sigma^2/D)$
Rev. water-fill	$D_i = \min(\sigma_i^2, \nu), \quad \sum D_i = D$
SLB	$R(D) \ge h(X) - \frac{1}{2}\log(2\pi e D)$

Binary $R(D)$	$h(p) - h(D), \quad 0 \le D \le p \le \frac{1}{2}$

C.5 Kolmogorov, Multi-User, Quantum

Object	**Formula / Property**
Invariance	$K(x) \le K_T(x) + c_T$
Coding thm	$K(x) = -\log P(x) + O(\log\log 1/P)$
Chain rule	$K(x,y) = K(x) + K(y\|x, K(x)) + O(\log K)$
Slepian-Wolf	$R_X \ge H(X\|Y)$, $R_X + R_Y \ge H(X,Y)$
Wyner-Ziv	$R^{WZ}(D) = \frac{1}{2}\log^+(\sigma^2_{X\|Y}/D)$
MAN caching	$R^*(t) = K(1 - t/N)/(1+t)$
Von Neumann	$S(\rho) = -\mathrm{tr}(\rho \log_2 \rho)$
Strong subaddit.	$S(ABC) + S(B) \le S(AB) + S(BC)$
Quantum Hamming	$\sum_{j=0}^{t} \binom{n}{j} 3^j \le 2^{n-k}$
Quantum capacity	$Q = \lim_n I_c(\mathcal{E}^{\otimes n})/n$
Normal approx.	$\log M^* = nC - \sqrt{nV}Q^{-1}(\varepsilon) + O(\log n)$